Rapport de stage
Calcul de chemins minimaux pour la détection de fissures

Gaël Michelin
2e année IN

28 septembre 2012

Remerciements

Je tiens tout d'abord à exprimer mes remerciements à Sylvie Chambon, qui m'a encadré durant tout mon stage, a répondu à mes nombreuses questions et m'a apporté beaucoup de conseils pour la rédaction de ce rapport.

Je souhaite également remercier Vincent Charvillat et Géraldine Morin, qui m'ont invité à réaliser mon stage dans leur équipe de travail, ainsi que l'ensemble de l'équipe Vortex pour leur accueil et leur disponibilité.

Mes remerciements s'adressent aussi à Xavier Descombes, qui m'a accordé de son temps et m'a proposé de nombreuses suggestions pour mon travail.

Enfin, je tiens à remercier Tien Sy Nguyen, dont les travaux de thèse ont constitué l'axe principal de mon stage, et qui a accepté de répondre à quelques-unes de mes questions au sujet de sa thèse.

Table des matières

Table des figures

Liste des algorithmes

Entreprise

Le laboratoire

L'IRIT [1] représente un des plus forts potentiels de la recherche en informatique en France avec un effectif global de 600 personnes dont 250 chercheurs et enseignants-chercheurs, 244 doctorants, 14 Post-Doctorants et chercheurs contractuels ainsi que 43 ingénieurs et administratifs. Le laboratoire est divisé en 19 équipes de recherche qui se structurent autour de sept thèmes scientifiques, qui constituent l'ensemble des domaines de l'informatique moderne :

- Analyse et synthèse de l'information
- Indexation et recherche d'informations
- Interaction, autonomie, dialogue et coopération
- Raisonnement et décision
- Modélisation, algorithmes et calcul haute performance
- Architecture, systèmes et réseaux
- Sûreté de développement du logiciel

Depuis quelques années, l'IRIT concentre ses recherches sur quatre axes stratégiques autour des grands défis scientifiques et socio-économiques :

- Informatique pour la santé, abordée au sein de l'IRIT selon quatre domaines de compétences : l'imagerie du vivant ; la gestion de données biomédicales et infrastructure d'accès et de traitement ; la modélisation et la simulation du vivant ; ainsi que le handicap et l'autonomie.
- Masses de données et calcul, qui concerne les différentes facettes du traitement et du calcul de données massives et couvre à la fois les infrastructures et les algorithmes.
- Systèmes sociotechniques ambiants, dont les composants peuvent être des entités physiques ou des logiciels distribués, qui ont des capacités d'interaction, un comportement autonome et ont la capacité de s'adapter à la tâche courante de l'être humain et aux ressources numériques et physiques disponibles.
- Systèmes embarqués critiques, conçus comme un assemblage de composants communiquant par un ou plusieurs réseaux numériques, dont on veut garantir des propriétés de qualité de service ; on étudie pour cela des environnements de développement certifiés et on développe des plateformes d'exécution.

1. Institut de Recherche en Informatique de Toulouse.

L'équipe

L'équipe VORTEX [2], au sein de laquelle j'ai réalisé l'ensemble de ce stage, est l'une des 19 équipes de recherche du laboratoire. Elle a été créée en 2006 afin de permettre le regroupement des équipes SIRV [3] et VPCAB [4]. VORTEX regroupe des compétences dans les domaines de l'informatique graphique, de la vision par ordinateur, de l'intelligence artificielle et des réseaux. La problématique principale de l'équipe est de visualiser des objets 2D ou 3D dans des environnements réels, virtuels ou mixtes. Ces objets peuvent être des images, des vidéos, des géométries 2D et 3D, ou encore des données plus complexes telles que des textures ou des entités évoluant de façon autonome au sein d'environnements virtuels.

VORTEX est principalement regroupée autour de trois axes de recherche :
- Acquisition, création et visualisation.
- Simulation comportementale et de la vie artificielle.
- Environnements virtuels et augmentés.

Mon stage s'inscrit dans le premier de ces axes : il s'agit d'extraire des informations d'une image de chaussée afin de prévenir les apparitions de fissures.

2. *Visual Objects : from Reality To EXpression.*
3. *Spherically Invariant Random Vectors.*
4. Vision Par Calculateur André Bruel.

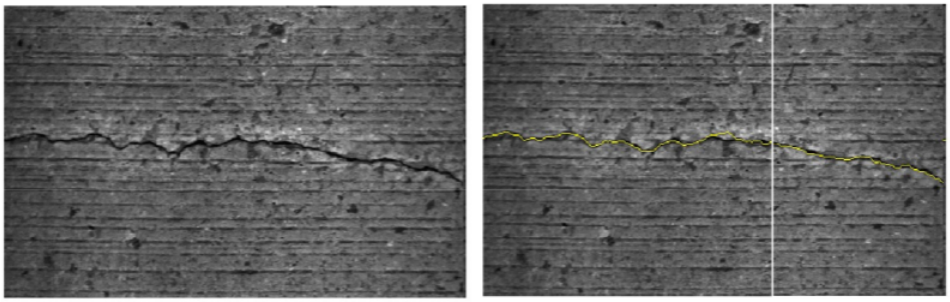
Contexte

Le relevé des dégradations des chaussées des routes nationales occupe une place très conséquente dans la gestion des réseaux routiers. En particulier, il est important pour les gestionnaires de pouvoir détecter le plus rapidement possible l'apparition de fissures sur les routes. Cependant, la très grande quantité de routes à contrôler rend nécessaire l'automatisation de cette tâche. Actuellement, la détection des fissures se fait en laboratoire, de façon manuelle, où des personnes relèvent d'éventuels défauts sur des images acquises sur le terrain.

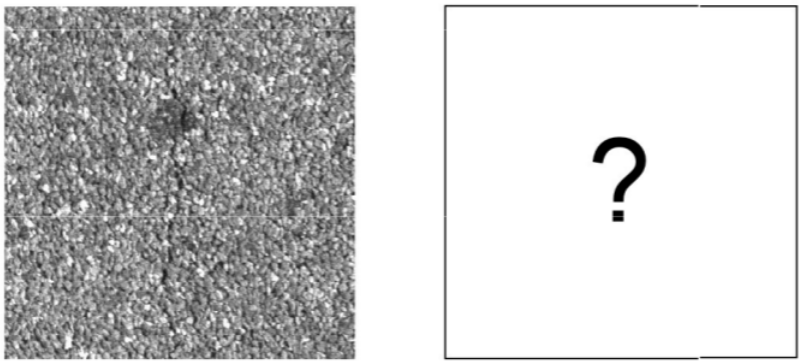
FIGURE 1 – Méthode actuelle de relevé de dégradations de surfaces de chaussées en France (illustration de [Ngu10]).

Face à cette méthode de gestion particulièrement coûteuse, de nombreux travaux sur la détection de défauts par traitement d'images ont été initiés. Toutefois, cette tâche s'avère délicate (figure 2). Dans ce travail, nous nous proposons d'étudier une méthode automatique de détection de ces défauts qui a été présentée par Nguyen [Ngu10]. En particulier, nous verrons la méthode dite de *Free Form Anisothropy* qui présente *a priori* des résultats satisfaisants. D'autre part, nous proposons d'étudier une approche Markovienne (déjà étudiée au lcpc [CM11]), en prenant en compte un algorithme tel que celui présenté dans [Lac04]. Notre objectif est d'implémenter ces approches et d'étudier leur comportement avec une base de données utilisées à l'IFSTTAR (*Institut Français des Sciences et Technologies pour les Transports, l'Aménagement et les Réseaux*).

Ce rapport est organisé de la manière suivante :



(a) Exemple d'une fissure sur la chaussée australienne et résultat de la détection par le système RoadCrack.
http://www.csiropedia.csiro.au/display/CSIROpedia/RoadCrack+system



(b) Exemple d'une fissure sur la chaussée française, bien plus texturée que la chaussée australienne.

FIGURE 2 – Exemples de fissures sur des chaussées différentes – Alors qu'il existe d'ores et déjà des systèmes performants de détection de fissures pour des chaussées présentant une texture semblable à la figure (a), la recherche actuelle ne propose pas de méthode suffisamment au point pour un revêtement tel qu'en (b), très présent en France.

- Dans une première partie, nous présentons la manière dont les fissures et les images sont modélisées et nous décrivons les différents principes de la méthode étudiée. En annexe, nous exposons également la manière dont s'utilise l'implémentation des différentes étapes de l'approche de [Ngu10].
- La seconde partie s'oriente sur une approche Markovienne du problème, de type RJMCMC (*Reversible Jump Markov Chain Monte Carlo*) [Lac04]. Une annexe consacrée à son utilisation se trouve également à la fin de ce rapport.
- La dernière partie du rapport aura pour objet l'analyse des résultats selon certains critères d'évaluation.

Chapitre 1

Approche proposée par [Ngu10]

1.1 Introduction

L'image est assimilée à une matrice de niveaux de gris allant de 0 (noir) à 255 (blanc). Une fissure sur cette image est supposée être une structure fine ayant une orientation aléatoire localement mais qui admet globalement une tendance selon une certaine direction ainsi qu'une relative continuité. On suppose enfin que le niveau de gris moyen d'un pixel situé sur une fissure sera plus faible que celui d'un pixel de fond.

L'hypothèse d'une fissure qui ait une certaine continuité et qui soit localement moins lumineuse que le fond de l'image peut nous conduire intuitivement à la notion de « chemin minimum » en terme de niveau de gris : un chemin situé le long d'une fissure semble posséder une luminance moyenne plus faible qu'un chemin quelconque du voisinage de cette fissure.

La mesure appelée *Free Form Anisothropy* [Ngu10] (FFA) est basée sur la recherche de ces chemins minimaux sur l'image d'étude selon des orientations données. Nous définirons dans ce chapitre la modélisation de l'image par des graphes orientés (8 graphes orientés par image), sur lesquels seront calculés des chemins minimaux en chaque point. La configuration des pixels qui constituent ces chemins pour chaque orientation permet alors le calcul de la mesure FFA en chaque point de l'image. Cette mesure traduit une probabilité d'appartenance d'un point à une fissure. Puis nous utiliserons la méthode d'Otsu [Ots79] pour réaliser une segmentation de l'image FFA obtenue. Enfin nous verrons comment réaliser l'opération de connexion des composantes connexes.

1.2 Notations

Dans la suite de ce rapport, les notations suivantes seront utilisées :
- I désigne l'image de l'étude,
- l désigne un pixel de l'image I,
- $I(l)$ désigne le niveau de gris du pixel l de l'image,
- N est le nombre total de pixels de l'image,

– d désigne la profondeur de calcul des chemins minimaux en chaque pixel de l'image. Chaque chemin minimal est de longueur $2d + 1$.

1.3 Modélisation de l'image

L'image peut être modélisée comme un graphe valué $G = (S, A)$ où chaque élément de S, l'ensemble des sommets, correspond à un pixel de l'image, et A définit l'ensemble des arêtes entre deux sommets.

Un voisinage traditionnel d'un pixel est le voisinage 4-connexe ou 8-connexe. Ici, nous utilisons des *voisinages directionnels* où chaque type de voisinage définit une orientation de graphe correspondante (voir la figure 1.1).

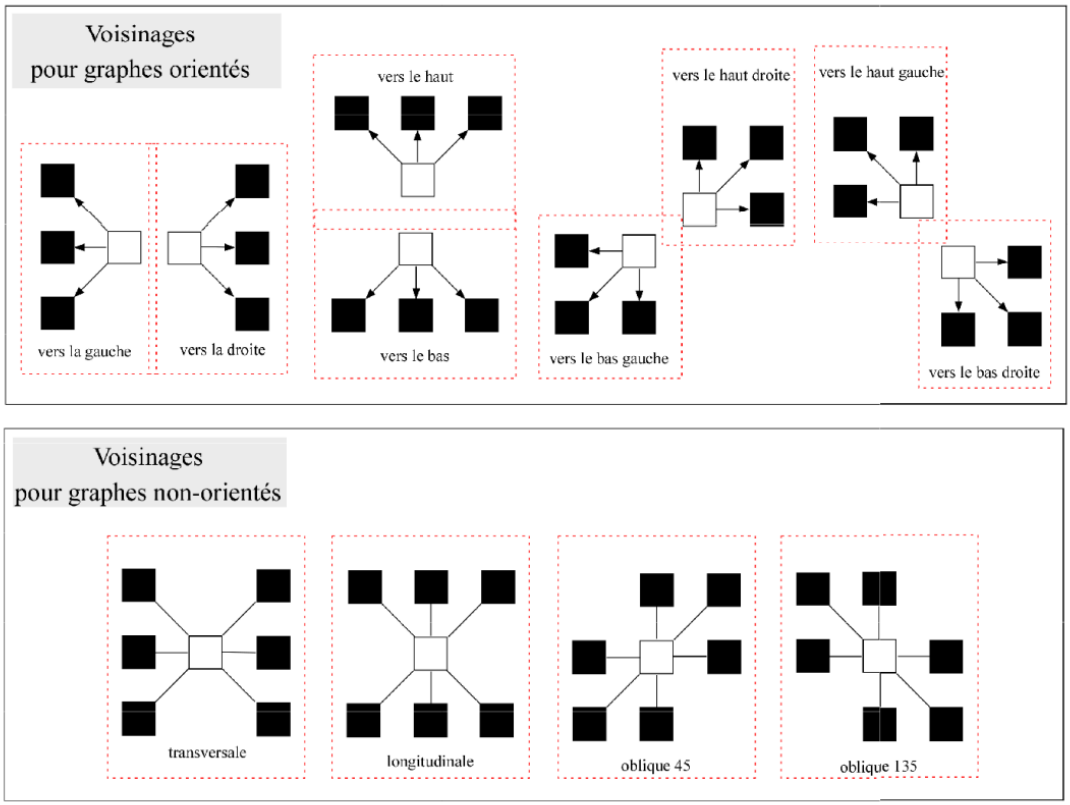


FIGURE 1.1 – Voisinages directionnels selon l'orientation et la direction (illustration issue de [Ngu10]).

Nous utiliserons les graphes orientés pour la recherche de chemins minimaux, alors que les graphes non orientés serviront à la représentation des chemins.

1.4 Modélisation de la fissure

Une fissure est modélisée comme un *chemin minimum* de longueur variable et de largeur 1 pixel. Pour détecter les fissures, nous sommes amenés à rechercher des fissures de longueur $1 \times d$ sur l'image. Ainsi, l'objectif est, pour tout pixel de l'image, de déterminer selon chaque orientation quel est le chemin de longueur fixée d partant de ce pixel et de valuation la plus faible, où la notion de valuation sera définie dans la section 1.6. De cette façon, le chemin minimum de longueur $2 \times d$ et de pixel central (i, j) sera pour chacune des 4 directions la concaténation des deux chemins orientés d'origine (i, j) (illustration figure 1.2).

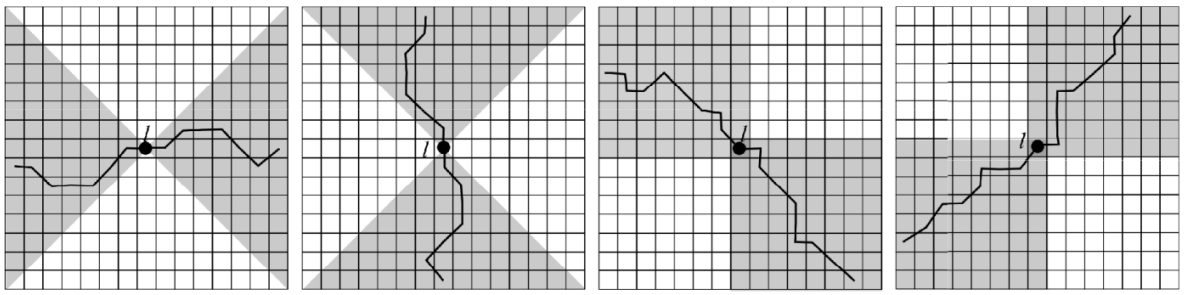


FIGURE 1.2 – Exemple de chemins minimaux partant de l pour les 4 directions possibles (illustration issue de [Ngu10]).

1.5 Étapes de l'approche

Les étapes de l'algorithme de [Ngu10] sont les suivantes :

1. Calcul des chemins minimaux selon 4 directions pour une certaine profondeur en chaque pixel de l'image de travail.
2. Calcul d'une « image FFA » à partir des caractéristiques des chemins minimaux en chaque pixel.
3. Binarisation de l'image FFA obtenue afin de déterminer les zones à forte probabilité de présence de fissure.
4. Décomposition en composantes connexes.
5. Connexion des composantes connexes détectées pour donner davantage de continuité au résultat trouvé.

1.6 *Free Form Anisotropy*

La valuation d'une distribution π est définie de la manière suivante :

$$val(\pi) = \sum_{l \in \pi} I(l)$$

Plus la valuation de π est faible, plus le niveau de gris moyen des pixels composant la distribution est petit. On définit également l'écart-type de π ainsi :

$$\sigma(\pi) = (\sum_{l \in \pi} (I(l) - \overline{I(l)}))^{1/2}$$

où $\overline{I(l)}$ est la moyenne des niveaux de gris des pixels de la distribution.

La distribution du chemin minimum de longueur $2d+1$ selon une direction dir en un pixel donné l est déterminée par :

$$\pi_{dir,d}(l) = \underset{\substack{\pi.centre=l \\ \pi.orientation=dir \\ \pi.taille=2d+1}}{argmin} (val(\pi))$$

Pour chaque pixel, à d fixé, 4 chemins minimaux sont donc déterminés selon les 4 directions possibles (voir figure A.2) et définissent 4 distributions correspondantes. L'hypothèse formulée précédemment permet d'exploiter ces résultats : si un pixel est situé sur une fissure, l'hypothèse de la taille fine et de la direction « stable » permet de considérer qu'il existera une direction parmi les 4 (transversale, longitudinale et obliques 45° et 135°) recouvrant cette fissure et que le parcours trouvé recouvrira bien la fissure. Le parcours correspondant sera de valuation la plus faible par rapport aux autres parcours trouvés (qui ne recouvriront pas une fissure).

Afin de détecter les fissures grâce à ces résultats, les auteurs s'appuient sur une nouvelle mesure : la « Free Form Anisotropy » ou FFA, définie ainsi :

$$FFA(l) = 1 - h(\pi_{min}, \pi_{fond})$$

où :

- $\pi_{min} = \underset{dir,d}{argmin}(\pi_{dir,d}(l))$ correspond à la distribution des pixels du parcours de valuation minimale au pixel l, pour une profondeur d fixée,
- $\pi_{fond} = \underset{dir,d}{argmax}(\pi_{dir,d}(l))$ est la distribution des pixels du parcours minimal de valuation maximale trouvé en ce pixel, pour une profondeur d fixée,
- h est une fonction traduisant le degré de cohérence entre deux distributions (1 : distributions très cohérentes, 0 : distributions non cohérentes). La valeur de $h(\pi_1, \pi_2)$ est fonction des valuations et des écarts-types de π_1 et π_2. Plus h sera petit, plus il sera probable que l suive une fissure.

Au terme de cette étape, nous obtenons une image FFA où la valeur d'un pixel traduit la probabilité que ce dernier appartienne à une fissure (probabilité d'autant plus forte que la valeur est élevée).

1.7 Algorithme de recherche du chemin minimum

La valuation d'une transition est égale au niveau de gris du pixel d'arrivée. Pour cette étude, nous appliquons une méthode incrémentale de complexité linéaire avec la taille d des chemins recherchés et le nombre N de pixels de l'image. L'idée est de déterminer les chemins minimaux de longueur k à partir de ceux de longueur $k-1$ sur l'ensemble de l'image. Il est nécessaire pour cela de définir

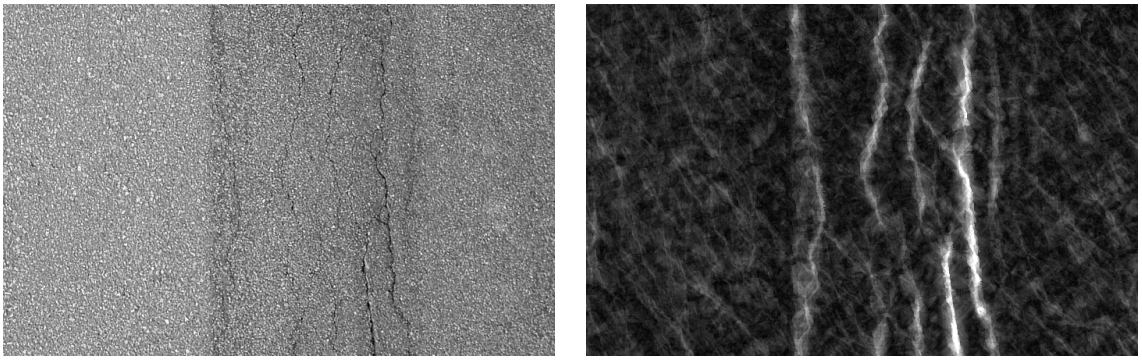


Image d'origine Image FFA

FIGURE 1.3 – Image FFA obtenue à partir d'une image de chaussée – On observe que les zones blanches sont situées sur ou à proximité de fissures. D'une façon peu attendue, on remarque que les zones non fissurées ont une valeur FFA assez variable, traduisant la présence de microfissures qui ne sont pas toujours facilement visibles à l'œil nu.

une structure de données associée à l'ensemble des sommets du graphe orienté qui soit adaptée au problème.

Pour la recherche de chemins minimaux de taille d sur un graphe orienté, on associe à chaque sommet les attributs suivants :

- value$[0..d]$
- voisin_min$[0..d]$

Le tableau value permet de mémoriser les valués des chemins minimaux (somme des niveaux de gris des pixels empruntés) d'origine le sommet et de longueur 0 à d, et voisin_min enregistre les positions du voisin correspondant au chemin minimum pour chaque longueur de chemin.

L'application de l'algorithme 1.1 sur les huit graphes orientés permet d'obtenir l'ensemble des chemins minimaux pour chaque pixel, pour chaque orientation et pour toute profondeur inférieure ou égale à d.

On note une influence de la valeur de d sur l'image FFA trouvée (figure 1.4). Plus la valeur de d est grande, plus les chemins minimaux calculés seront grands et leur position de départ exacte sera moins déterminante dans le calcul de la valeur FFA, ce qui se traduit par un élargissement des zones autour des fissures. Une valeur de d plus faible délimite de façon plus nette les zones de fissures, mais un choix de d trop faible ne permettra pas de différencier une véritable fissure d'une microfissure dans la texture de l'image.

1.8 Binarisation

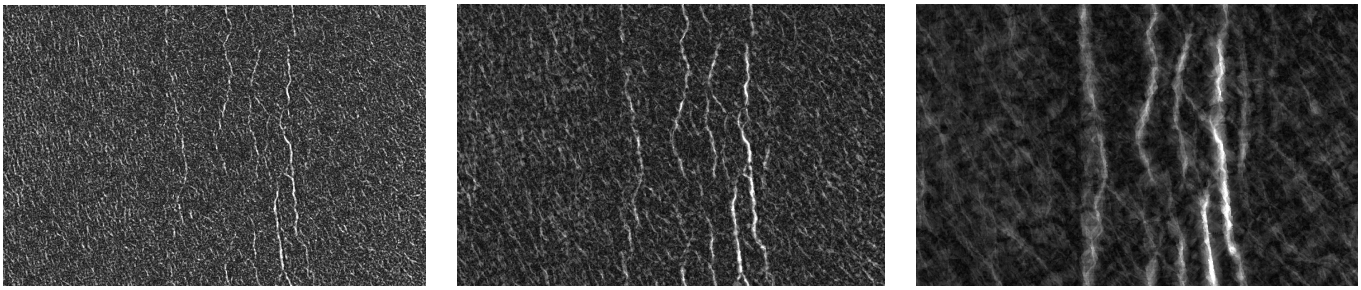
Une fois les chemins minimaux estimés en chaque pixel, on effectue le calcul de l'image FFA. Chaque pixel a une valeur comprise entre 0 et 1 mais on souhaite donner une réponse binaire (présence ou non de fissure) et c'est la raison pour laquelle on effectue une binarisation de l'image. Pour cela, une

Algorithme de recherche du *chemin minimum*

Données :
$G =< S, A >$=Graphe orienté et valué modélisant l'image traitée pour une orientation donnée.
s_l^k désigne le sommet correspondant au pixel l à la k^{ime} itération de l'algorithme.
Résultat :
On obtient les chemins minimaux de longueur $0..d$ et leur valuation pour chaque sommet du graphe qui sont stockés dans S^d avec $d \in [0; d]$

Début d'algorithme
Initialiser $S^0 = \{s_l^0 | l$ pixel de l'image$\}$:
$s_l^0.\,\text{value}[0] = I(l)$
$s_l^0.\,\text{voisin\_min}[0] = s_l^0$
Pour *longueur* $k = 1$ à d **faire**
 Pour chaque sommet $s_l^k \in S^k$ **faire**
  Chercher le voisin $v_{s_l} \in \nu(s_l)$ qui a le valué minimum de chemin minimum
  d'origine vs_l de longueur $k-1$:
  $v_{s_l,min} = \underset{vs_l}{argmin}(v_{s_l}.\,\text{value}[k-1])$
  Mettre à jour s_l^k : $s_l.\,\text{value}[k] = l + v_{s_l,min}.\,\text{value}[k-1]$
  $s_l.\,\text{voisin\_min}[k] = v_{s_l,min}$
 Fin
Fin
Fin d'algorithme

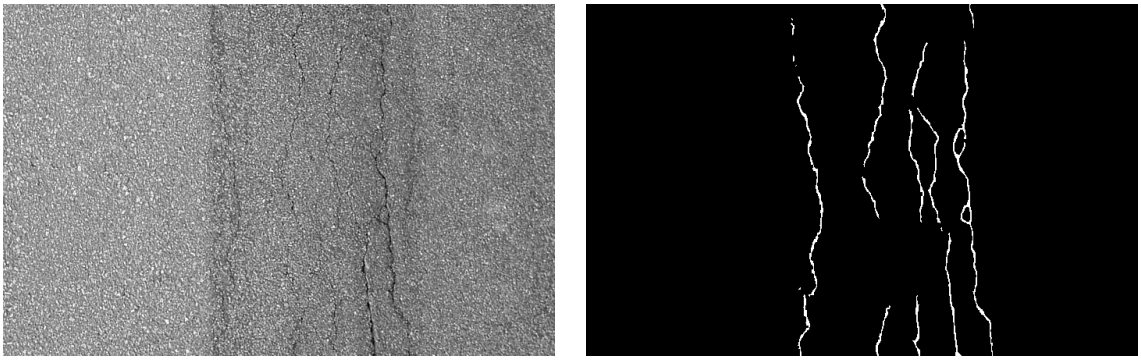
ALGO. 1.1 – *Méthode incrémentale – La complexité de cet algorithme est en $\mathcal{O}(d \times N)$ où N est le nombre de pixels de l'image.*



$d = 5$ $d = 10$ $d = 30$

FIGURE 1.4 – Influence de d sur l'image FFA – Les microfissures présentes sur l'image d'origine ressortent moins sur l'image FFA lorsque d croît. En contre-partie, les zones connexes à la fissure sont alors moins affinées que pour un d plus faible car la valeur FFA calculée est d'autant moins sensible à la position exacte du pixel central que d est grand.

approche de seuillage à deux niveaux a été choisie.



(a) (b)

FIGURE 1.5 – (a) Image de test, (b) Vérité de terrain de l'image.

1.8.1 Méthode par seuillage à deux niveaux

Cette étape est réalisée en effectuant un **seuillage à deux niveaux**. On détermine pour cela un seuil haut au dessus duquel un pixel est considéré comme élément d'une fissure et un seuil bas en dessous duquel un pixel est considéré comme un pixel de fond. Les pixels compris entre ces deux seuils sont alors considérés comme appartenant à une fissure si et seulement s'ils sont connexes à un pixel de fissure par un chemin passant par des pixels supérieurs au seuil bas.

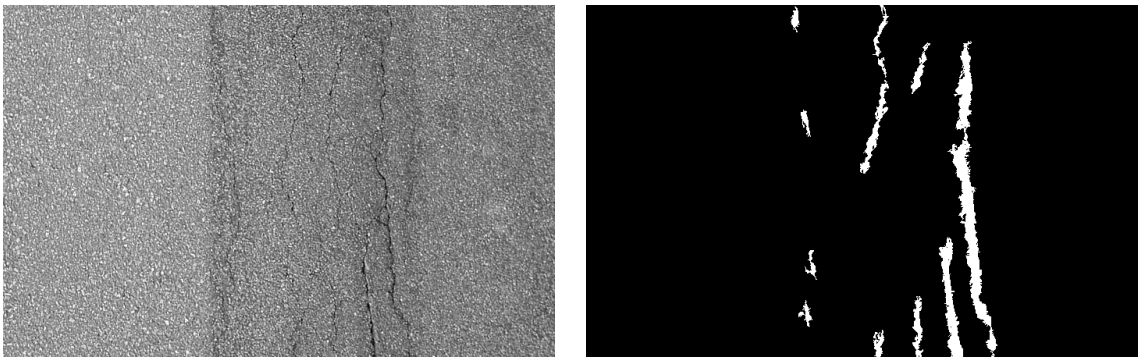


FIGURE 1.6 – Résultat de la binarisation de l'image FFA ($d = 30$).

La principale difficulté réside dans la détermination automatique des seuils haut et bas optimaux pour une image FFA donnée. Nous appliquons pour cela la méthode d'Otsu [Ots79]. Les deux seuils définissent trois classes $C_0 = \{0..t_1\}$, $C_1 = \{t_1..t_2\}$ et $C_2 = \{t_2..L\}$ où $L = 255$ et le but est de maximiser la variance inter-classe totale, *i.e.*

$$\{t_1^*, t_2^*\} = \underset{0<t_1<t_2<L}{argmax} (\sigma_B^2(t_1, t_2)) \tag{1.1}$$

où

$$\sigma_B^2 = \sum_{k=0}^{2} \omega_k (\mu_k - \mu_T)^2 \tag{1.2}$$

avec

- $p_i = n_i/N$ est la probabilité de distribution du niveau i, où n_i est le nombre de pixels de niveau i,
- $\omega_k = \sum_{i \in C_k} p_i$ est la probabilité de la classe k,
- $\mu_k = \frac{1}{\omega_k} \sum_{i \in C_k} i p_i$ est la moyenne des pixels de la classe k,
- $\mu_T = \sum_{k=0}^{2} \omega_k \mu_k$ est la moyenne des pixels de toute l'image FFA.

1.8.2 Choix d'implémentation

Détermination des seuils

La méthode de binarisation que nous présentons nécessite d'utiliser des seuils appropriés à l'image FFA traitée. Pour déteminer ces seuils, nous appliquons une alternative à la méthode d'Otsu [Ots79] (*Image thresholding*), le *Fast Algorithm for Multilevel Otsu's Method* [LCC01]. En reprenant les notations vues en section 1.8.1 (page 11), on rappelle les propriétés suivantes

$$\sum_{k=0}^{2} \omega_k = 1 \tag{1.3}$$

$$\mu_T = \sum_{k=0}^{2} \omega_k \mu_k \tag{1.4}$$

Ces propriétés utilisées dans la formule (1.5) établissent que la variance inter-classes de l'image seuillée peut s'écrire

$$\sigma_B^2 = \sum_{k=0}^{2} \omega_k \mu_k^2 - \mu_T^2 \tag{1.5}$$

Comme μ_T est indépendante du choix des seuils, la minimisation de σ_B^2 revient à minimiser $(\sigma'_B)^2$ où

$$(\sigma'_B)^2 = \sum_{k=0}^{2} \omega_k \mu_k^2 \tag{1.6}$$

Finalement, la formulation du problème (1.1) devient

$$\{t_1^*, t_2^*\} = \underset{0<t_1<t_2<L}{argmax} \left((\sigma'_B)^2(t_1, t_2)\right) \tag{1.7}$$

avec t_1^* et t_2^* les seuils bas et haut recherchés.

L'application directe de cette approche débouche cependant sur un résultat inexploitable (figure 1.8) car les seuils sont calculés sans *a priori* sur le résultat à trouver, et l'histogramme des niveaux de gris de l'image (figure 1.7) explique que les seuils calculés soient trop faibles.

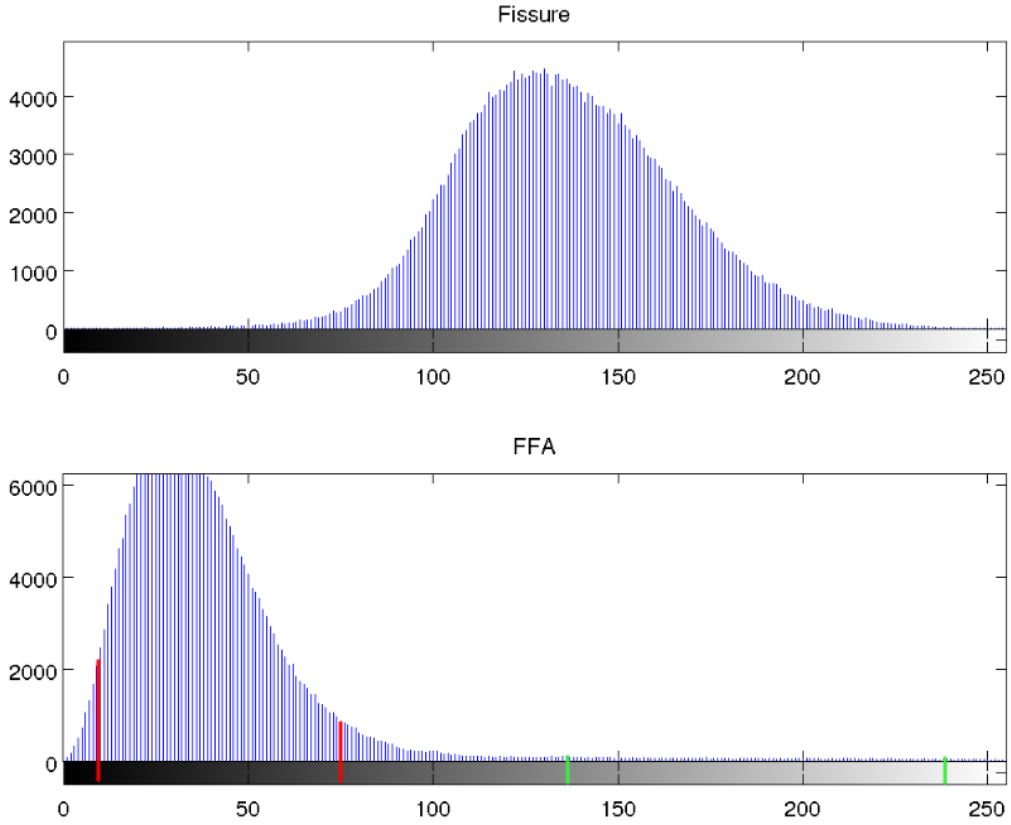


FIGURE 1.7 – Histogrammes de l'image originale et de l'image FFA – En rouge, les seuils calculés par la méthode directe d'Otsu. En vert, les seuils calculés par l'approche avec *a priori*.

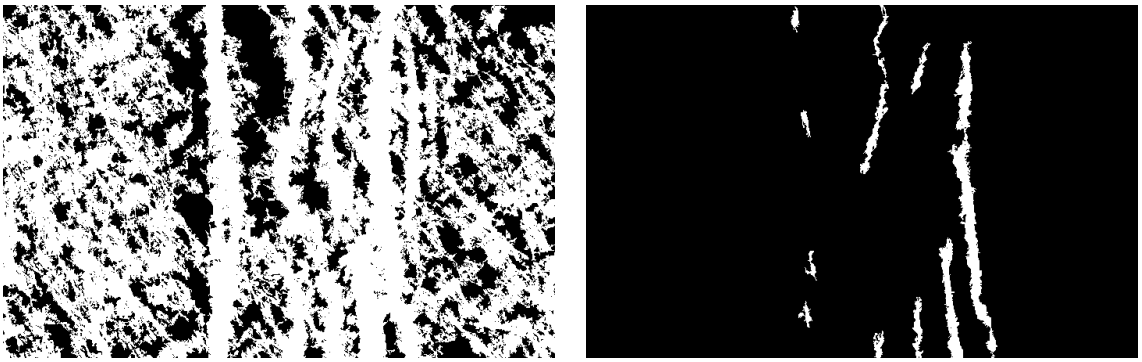
Il est donc apparu nécessaire de modifier les bornes des seuils à partir desquelles nous appliquons la détection automatique des seuils. Partant du principe qu'une fissure ne recouvre que 1 à 2% d'une image, on pose comme limite inférieure de $seuil_B$ la première valeur de k telle que :

$$\sum_{i=0}^{k} p_i > \alpha \tag{1.8}$$

où α détermine le taux de détection de fissure maximal sur l'image : pour un choix de $\alpha = 0,97$, 3% des pixels de l'image auront une valeur FFA supérieure à $seuil_B$ et seront donc potentiellement détectables. Cependant, le taux de recouvrement effectif n'est pas forcément aussi élevé et, en fonction de l'image, est plutôt de l'ordre de 2% pour $\alpha = 0,97$. En partant de l'hypothèse qu'une fissure ne peut généralement pas occuper plus de 1 à 2% de l'image, le choix de $\alpha = 0,97$ permet d'obtenir des résultats globalement satisfaisants.

On notera que par la méthode de binarisation proposée par [Ngu10], toute image sera détectée comme fissurée. Le choix des seuils bas et haut uniquement en fonction de l'histogramme de l'image FFA n'est donc pas complètement satisfaisant. Il plutôt intéressant de faire un apprentissage de ces valeurs de seuil sur une base de donnée d'images classifiées dans différentes familles en fonction du type de chaussée.

Une fois le $seuil_B$ minimal déterminé, nous appliquons alors normalement la méthode d'Otsu et obtenons des résultats exploitables.



(a) (b)

FIGURE 1.8 – Seuillage avec la méthode d'Otsu et la méthode modifiée – (a) Seuils déterminés avec application simple d'Otsu, (b) Seuils déterminés par la méthode d'Otsu après recherche du $seuil_B$ min.

Seuillage à deux niveaux

L'étape précédente nous fournit deux seuils, $seuil_B$ et $seuil_H$, où $seuil_B < seuil_H$. Nous réalisons alors les opérations suivantes (figure 1.9) :

- Les pixels inférieurs à $seuil_B$ sont mis à 0.
- Les pixels supérieurs à $seuil_H$ sont mis à 255.
- Les pixels de valeur intermédiaire sont mis à 255 s'il existe un chemin de pixels $> seuil_B$ les reliant à un pixel $> seuil_H$.

1.9 Connexion de composantes connexes

Les résultats obtenus présentent fréquemment des discontinuités, ce qui se traduit par une détection incomplète des fissures (voir la figure 1.9). La phase de connexion des composantes connexes doit permettre d'obtenir la forme entière des fissures. La méthode pour connecter les éléments de fissure est la suivante :

1. Pour chaque composante connexe cc_i, rechercher les points extrémités, où un point extrémité est un pixel l tel que son voisin suivant le chemin minimal d'origine l est un pixel n'appartenant pas à cc_i.
2. Un point extrémité l_i de cc_i sera connecté à une composante connexe cc_j s'il existe l_j extrémité de cc_j et respectant les conditions suivantes :
   - l_i et l_j ont des directions locales compatibles (par exemple si les parcours de l_i et l_j sont respectivement transversal et oblique 45°, les directions seront considérées comme compatibles ; si ces parcours sont transversal et longitudinal, les directions seront considérées comme contraires donc incompatibles).

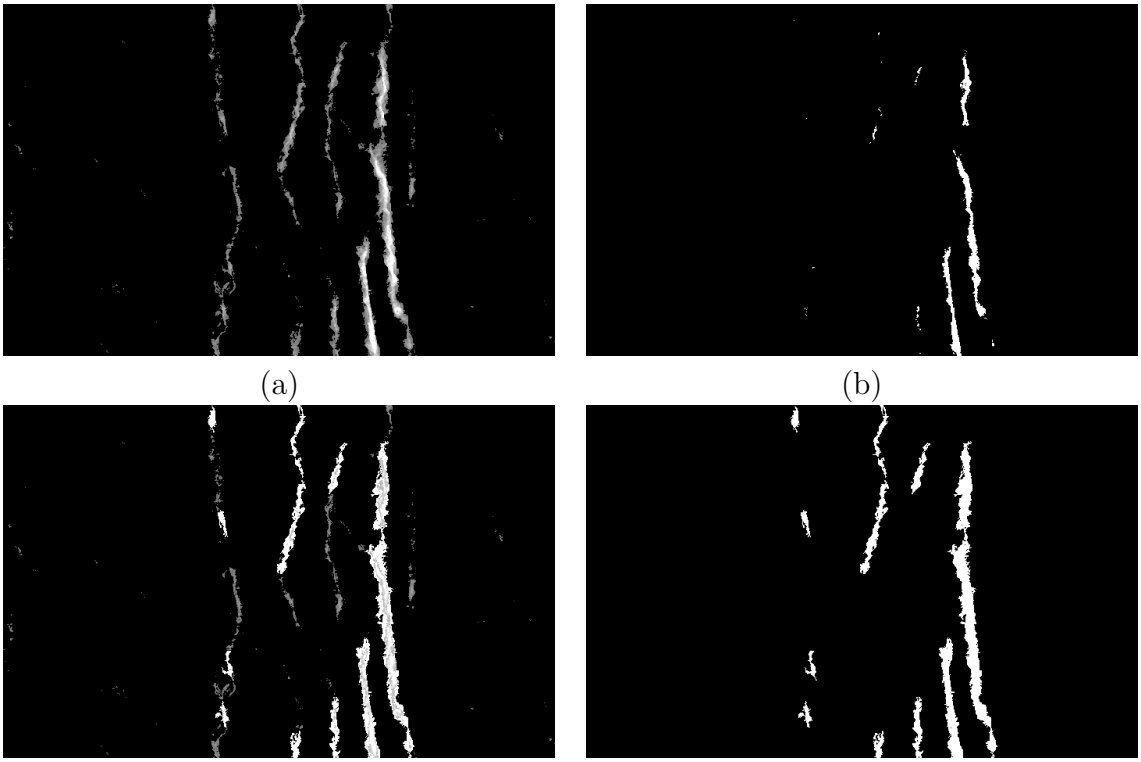


FIGURE 1.9 – Étapes de binarisation de l'image FFA – (a) Initialisation du seuillage bas , (b) Initialisation du seuillage haut, (c) Seuillage après ajout des composantes connexes, (d) Binarisation de l'image FFA.

– Le chemin min d'origine l_i de longueur d contient des pixels $\in cc_j$.
– Le chemin min d'origine l_j de longueur d contient des pixels $\in cc_i$.
Si les conditions précédentes sont respectées, l_i sera connecté à cc_j par le chemin minimum calculé en ce point.

Ainsi, en reprenant l'image qui nous sert d'exemple, on peut illustrer le type de résultat obtenu grâce à la figure 1.10.

1.10 Conclusion

La méthode FFA nécessite de choisir les paramètres suivants :
- d, la profondeur de calcul des chemins minimums.
- $seuil_B$ et $seuil_H$, les seuils permettant la binarisation de l'image, calculés par la méthode d'Otsu [Ots79].
- α, le pourcentage qu'on se donne pour fixer la valeur de $seuil_B$ (section 1.8.2).

Ce faible nombre de paramètres rend cette méthode relativement simple à calibrer pour une image donnée, mais on s'apperçoit qu'elle est limitée. En effet, une image trop fortement texturée (figure 1.11) induira en erreur la méthode FFA qui ne différenciera pas une véritable fissure d'une microfissure qu'on

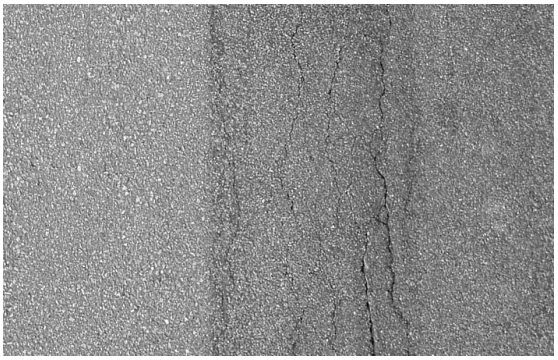


Image de la fissure

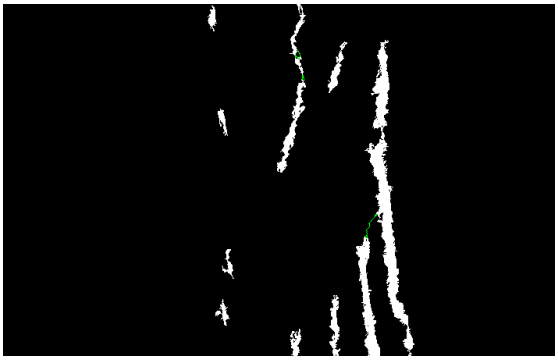
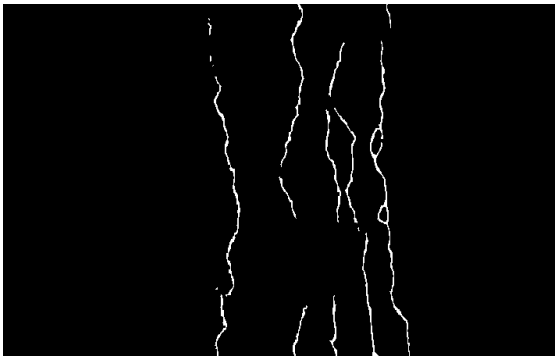


Image obtenue après connexion



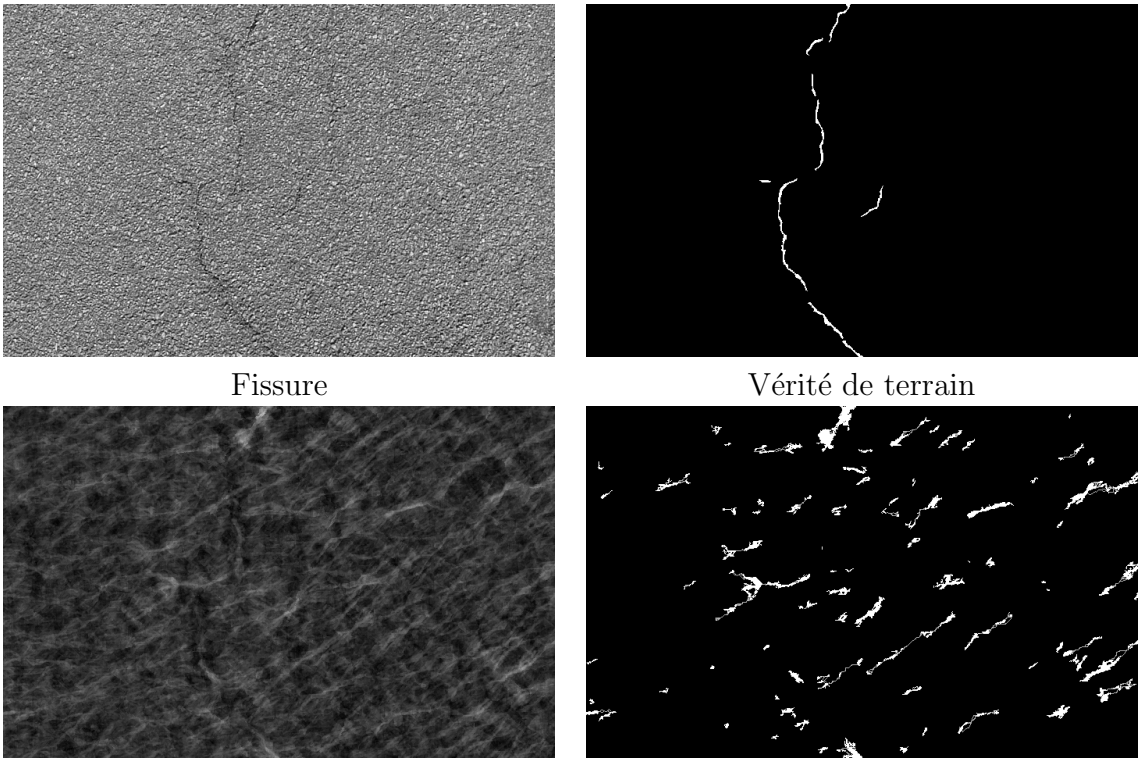
Vérité terrain associée à l'image

FIGURE 1.10 – Image binaire après la connexion – Les connexions sont représentées en vert et la profondeur $d = 30$.

ne cherche pas à détecter.

Nous reprendrons l'analyse des résultats obtenus avec cette approche dans le chapitre 3.

On notera toutefois que par la méthode de binarisation proposée par [Ngu10], toute image sera détectée comme fissurée. Le choix des seuils bas et haut uniquement en fonction de l'histogramme de l'image FFA n'est donc pas complètement satisfaisant. Il plutôt intéressant de faire un apprentissage de ces valeurs de seuil sur une base de donnée d'images classifiées dans différentes familles en fonction du type de chaussée.



Fissure | Vérité de terrain

Image FFA | Résultat post-connexion

FIGURE 1.11 – Cas d'une fissure très mal détectée par la méthode FFA.

Chapitre 2

Approche Markovienne : méthode RJMCMC [Lac04]

2.1 Introduction

Le résultat que nous avons obtenu avec la méthode *FFA* après binarisation aboutit à des résultats qui déterminent des zones dans lesquelles se situent potentiellement les fissures de l'image. Cette méthode par binarisation n'est pas pleinement satisfaisante, on observe que les zones détectées sont parfois bien plus épaisses qu'attendu et recouvrent en grande partie des pixels de fond.

Notre souhait était donc d'exploiter d'une façon différente l'image *FFA*. Cette image nous donne une connaissance *a priori* sur la probabilité qu'a un pixel d'être situé sur une fissure. On a également observé que les chemins partant de points situés sur des fissures suivent effectivement ces fissures. C'est pourquoi nous avons décidé d'implémenter la méthode RJMCMC (*Reversible Jump Markov Chain Monte Carlo*) en collaboration avec Xavier Descombes. Cette méthode repose sur une technique d'échantillonnage aléatoire et permet de modéliser l'image observée par une collection d'objets représentant les fissures.

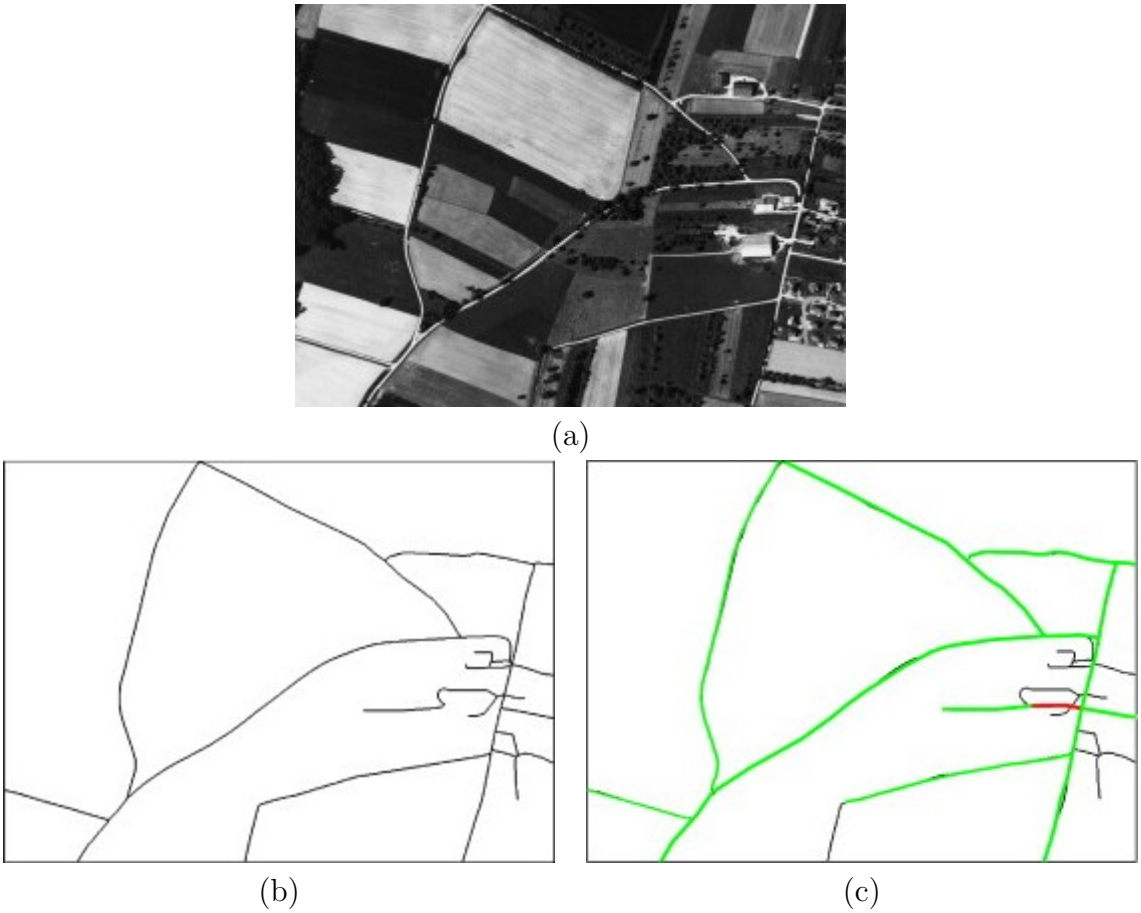
2.2 Méthode RJMCMC

2.2.1 Principe

Cette méthode repose sur la théorie des champs de Markov, comme l'ensemble des algorithmes de type Monte Carlo par Chaîne de Markov (MCMC). De plus amples informations sur la théorie de ces algorithmes peuvent être trouvées dans le document de thèse rédigé par Caroline Lacoste [Lac04].

La méthode RJMCMC est un modèle de processus ponctuels marqués qu'on utilise pour extraire des réseaux de formes, par exemple des réseaux de routes sur des images satellitaires, ou encore des colonies de flamants roses prises par photographies aériennes [BHCDZ10] (figure 2.1).

On modélise ce qu'on souhaite détecter par des objets, comme des ovales pour les flamants roses ou



(a) (b) (c)

FIGURE 2.1 – Méthode RJMCMC appliquée à la détection de routes (illustration issue de [LDZ09])– (a) Image satellitaire , (b) Vérité terrain de l'image, (c) Résultat obtenu par la méthode RJMCMC. En vert, les détections correctes. En rouge, les détections incorrectes.

des rectangles fins pour les portions de routes. Le but est de rechercher une configuration d'objets sur l'image qui retranscrive le plus fidèlement possible la réalité du terrain. Pour cela, on associe à chaque objet une **énergie** qui doit être positive dans le cas d'un objet mal positionné et négative si l'objet est correctement placé. Ainsi, il s'agira de trouver une configuration minimisant l'énergie globale des objets.

2.2.2 Énergie

L'énergie de chaque objet est la somme d'un terme d'**attache aux données** D (par exemple prenant en compte la luminance moyenne à l'intérieur de l'ovale pour la détection de flamants roses) et d'un terme d'**interactions** F avec les autres objets de la configuration (dans le cas des flamants roses, on cherchera à éviter les superpositions d'objets en leur associant une énergie positive pénalisante). Ainsi, l'énergie associée à une configuration d'objets E s'écrit :

$$U(E) = \sum_{o \in E} D(o) + \sum_{o \sim o'} F(o, o') \tag{2.1}$$

2.2.3 Noyaux de propositions

La configuration de départ de l'algorithme est la configuration vide. À chaque itération, partant d'une **configuration** x, on choisit parmi plusieurs **noyaux de propositions** notés $Q_m(x, .)$ celui qui sera réalisé, à partir des **probabilités de réalisation** $p_m(x)$. On retrouve communément parmi ces noyaux les processus de naissance et de mort d'objets, mais aussi des noyaux qui transforment un objet déjà existant (déplacement, rotation ou encore dilatation d'un objet). Les changements ainsi proposés induisent une **différence d'énergie** ΔU avec la précédente configuration, qui peut être soit positive soit négative. Une différence négative semble indiquer que la nouvelle configuration est meilleure que la précédente, alors qu'une différence positive est plutôt à éviter. Au terme de chaque itération, on choisit de conserver avec une certaine **probabilité d'acceptation** R_m le changement proposé, où la probabilité est d'autant plus forte que la différence d'énergie est négative. Ainsi, avec un nombre d'itérations suffisamment grand, on doit s'approcher de la configuration optimale.

Une condition de convergence de la méthode est qu'il doit être possible pour toute proposition de changement de configuration de x vers y de pouvoir proposer le changement de configuration opposé, de y vers x. La **probabilité de passage** de x vers y est notée $q(x, y)$ et celle de y vers x s'écrit $q(y, x)$.

2.2.4 Algorithme

Algorithme RJMCMC
Initialisation – Partir de la configuration vide. – Fixer les probabilités de tirer les différents noyaux de proposition. – Température initiale $T = T_0$. – Fixer le coefficient α de mise à jour de la température.
Itération en partant d'une configuration x – Choisir un noyau de proposition $Q_m(x, .)$ avec la probabilité $p_m(x)$, ou bien laisser l'état inchangé avec une probabilité $1 - \sum_m p_m(x)$. – Simuler l'état y suivant le noyau de proposition choisi. – Calculer le rapport de Green (ratio d'acceptation) : $R_m = exp(-\frac{\Delta U}{T}) \times \frac{q(x,y)}{q(y,x)}$, où $q(x_1, x_2)$ mesure la probabilité de passer de la configuration x_1 à la configuration x_2. – Avec une probabilité $p = min(1, R_m)$, accepter la proposition y, sinon conserver x. – Mise à jour de la température : $T \leftarrow \alpha T$. Réitérer jusqu'à convergence.

Algo. 2.1 – *Algorithme RJMCMC.*

Dans l'algorithme présenté dans l'encadré 2.1, on notera la présence d'une température initialement élevée et qui décroit lentement après chaque itération suivant un coefficient α. La différence d'énergie est d'autant plus discriminante que la température est élevée dans le calcul du taux d'acceptation du changement. Lorsque la température est nulle, le changement proposé est conservé s'il fait diminuer

l'énergie de la configuration et il est rejeté sinon. Quand la température est élevée, cela a pour effet d'amoindrir l'influence de ce changement d'énergie et de favoriser une modification qui présenterait un $\Delta U > 0$.

L'intérêt de commencer avec un T_0 élevé est de réaliser un *recuit simulé* qui permet d'obtenir une configuration qui soit un minimum global du problème, et pas simplement local.

Le choix des probabilités p_m de tirage des différents noyaux ne doit pas être déterminant pour le résultat. Ce choix s'effectue dans le but d'avoir une vitesse de convergence de l'algorithme optimale. Un **critère d'arrêt** commun de cet algorithme est d'effectuer un certain nombre (de l'ordre de la dizaine de fois le nombre d'objets de la configuration) d'**itérations à température nulle** et d'observer si une des propositions a été acceptée. Si oui, alors l'algorithme n'a pas convergé. Si non, l'algorithme prend fin.

2.3 Application à la détection de fissures

2.3.1 Problématique

L'objectif que nous nous étions fixé était d'exploiter les résultats calculés avec la méthode de [Ngu10]. Il s'agissait donc de trouver des objets ainsi que des énergies adéquates. Le calcul de l'image FFA passe par celui des chemins minimaux, qui suivent très fidèlement les fissures dans le cas où on positionne leur centre correctement. Nous avons donc choisi d'utiliser ces chemins minimaux précédemment calculés comme objets pour notre calcul de configurations. Nous appellerons dans la suite de ce rapport ces objets des *portions de fissure*. Les portions de fissure sont définies par un centre ainsi qu'une profondeur. La direction et le chemin suivis par la portion sont ensuite déterminés par ces seuls paramètres.

L'algorithme (encadré 2.1) nécessite ensuite de choisir des énergies d'attache aux données ainsi que d'interaction deux à deux entre les objets. Ce point a constitué l'une des principales difficultés de ce travail car il s'agit de parvenir à trouver des connaissances *a priori* sur la forme des fissures et de déterminer les relations souhaitées entre les différentes portions de fissures puis à doser les diverses influences les unes par rapport aux autres afin d'obtenir des résultats qui soient satisfaisants.

2.3.2 Terme d'attache aux données

Les valeurs FFA calculées auparavant sont une mesure de la probabilité en chaque pixel d'appartenir à une fissure. Cette valeur est déjà une mesure contenant une information sur l'environnement du pixel. On rappelle que plus la valeur FFA est élevée en un point, plus il a de chances d'être situé sur une fissure. Ainsi, une connaissance *a priori* sur une portion de fissure peut être apportée par la valeur FFA prise par le pixel central de cette portion. Le résultat obtenu sera une première indication sur la probabilité d'appartenance de la portion à une fissure. Il s'agit ensuite d'exploiter cette valeur pour la convertir en énergie, une énergie qui sera négative si la portion semble bien positionnée et positive sinon. Nous avons ainsi utilisé une fonction classique pour ce genre de problème. Si o désigne une portion de fissure, on a donc :

$$FFA(o) = FFA(centre(o)) \tag{2.2}$$

$$D(o) = 1 - \frac{2}{1 + exp\left(-\gamma\left(\frac{FFA(o)}{S} - 1\right)\right)} \tag{2.3}$$

Cette expression fait intervenir les constantes S et γ (figure 2.3).
- S est le seuil qui détermine la valeur limite de $\int FFA(o)$ au dessus duquel $D(o) > 0$ et en dessous duquel $D(o) < 0$.
- γ est un coefficient qui fait varier la « pente » de la fonction.

Une variante possible est de prendre comme connaissance *a priori* sur la portion l'intégrale normalisée des valeurs FFA prises par pixels le long de son chemin. Cette idée est cependant discutable car la valeur FFA en chaque pixel est une mesure calculée sur l'ensemble du chemin d'origine ce pixel, et donc prendre la somme des valeurs FFA le long d'une portion donnerait une grandeur qui n'est pas propre à la seule portion considérée mais aux $2d + 1$ portions d'origine les pixels de ce chemin.

2.3.3 Interactions

La question la plus délicate est de déterminer les énergies d'interactions. Le problème est double : **éviter les superpositions** de portions de fissures, mais aussi **favoriser les connexions** des portions entre elles, partant de l'hypothèse qu'une fissure est une structure fine et généralement allongée. Pour cela, le choix s'est finalement orienté vers l'ajout de deux termes d'interactions : un terme de **proximité** et un terme de **connexion**.

Le **terme de proximité** est un terme qui doit être pénalisant. Si o et o' sont deux portions, on note $P(o, o')$ leur énergie de proximité (figure 2.2).

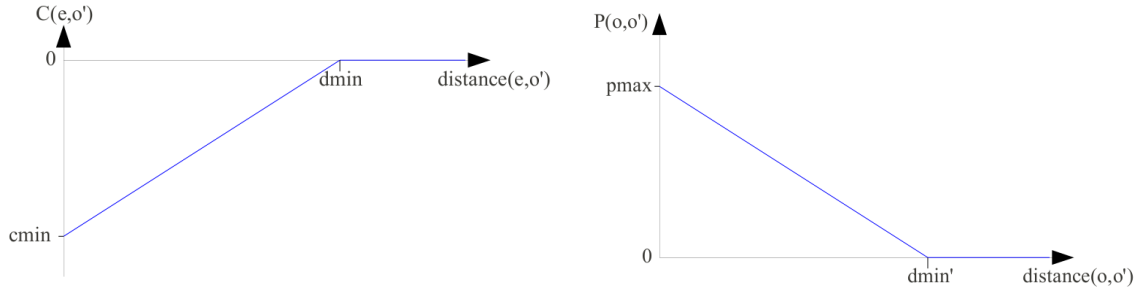
Un **terme de connexion** négatif $C(e, o')$ récompense le fait que l'extrémité e d'une portion o soit proche d'une portion o' (figure 2.2) . Si $C(o, o')$ est l'énergie de connexion entre o et o', on a :

$$C(o, o') = C(e_1, o') + C(e_2, o') + C(e'_1, o) + C(e'_2, o) \tag{2.4}$$

Le choix du type de fonction qui régit ces énergies n'était pas fondamentalement important. Une fonction affine par morceaux (figure 2.2) était suffisante pour obtenir l'influence voulue sur les résultats.

L'énergie de connexion $C(.,.)$ fait apparaître **deux nouvelles constantes**, $c_{min} < 0$ qui correspond à l'énergie produite dans le cas d'une connexion réalisée entre deux portions, et d_{min} telle que si une extrémité est à une distance $d \geq d_{min}$ d'un objet, l'énergie de connexion induite est nulle, sinon elle est négative.

L'énergie de proximité $P(.,.)$ fait quant à elle intervenir **les constantes** $p_{max} > 0$ et d'_{min} où d'_{min} est la distance limite entre les centres de deux portions en dessous de laquelle il y a une influence pénalisante pour la configuration qui vaut au plus p_{max} si la distance est nulle, et au dessus de laquelle il n'y a pas de pénalisation.



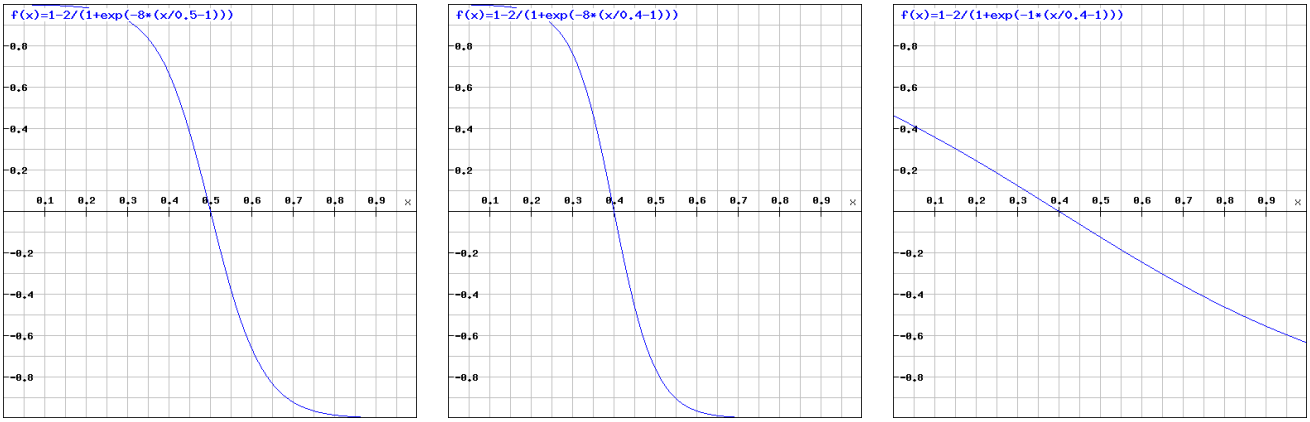
(a) (b)

FIGURE 2.2 – Énergies d'interaction – (a) Énergie de connexion entre l'extrémité e d'un objet o et l'objet o', (b) Énergie de proximité entre o et o'.

2.4 Choix des paramètres

2.4.1 Attache aux données

L'équation (2.3) fait intervenir les constantes S et γ qu'il faut fixer. Les différents résultats nous



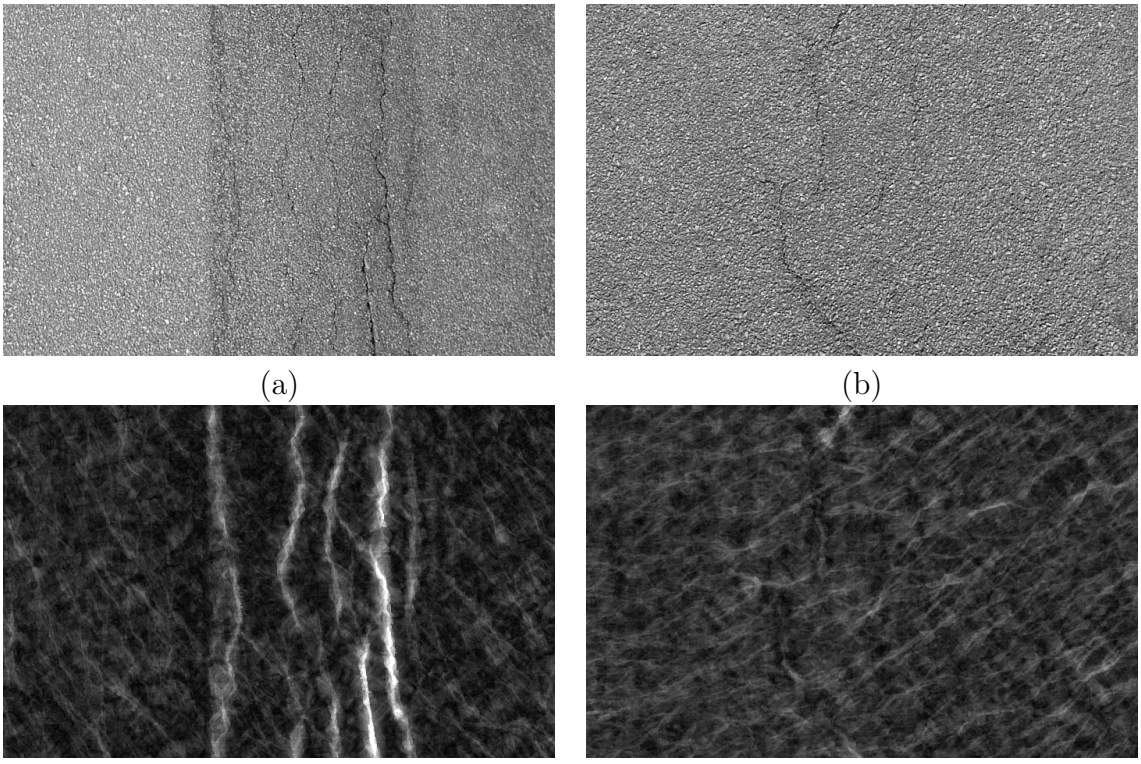
(a) (b) (c)

FIGURE 2.3 – Influence de S et γ sur $D(o)$ – (a) $S = 0.5$, $\gamma = 8$, (b) $S = 0.4$, $\gamma = 8$, (c) $S = 0.4$, $\gamma = 1$.

ont amenés à prendre une valeur de γ suffisamment élevée pour que l'énergie d'attache aux données soit discriminante, tout en gardant une évolution de l'énergie progressive. Finalement, $\gamma = 8$ était un bon compromis que nous avons conservé dans la suite de ces travaux.

Le choix du seuil S est en revanche sujet à davantage de discussions. Un seuil trop faible engendrerait une surdétection de fissures alors qu'un seuil trop élevé en détecterait trop peu. Ce choix de seuil est d'autant plus complexe qu'il dépend de l'image FFA dont la qualité est très variable en fonction du type de chaussée (figure 2.4).

L'influence de S sur le résultat est illustré par la figure 2.5.



(a) (b) (c) (d)

FIGURE 2.4 – Images FFA pour différentes chaussées – (a) et (b) échantillons de chaussée, (c) et (d) leur image FFA associée. Alors que l'image FFA (c) permet la mise en évidence de la fissure, l'image FFA (d) ne le permet pas.

2.4.2 Interactions

La difficulté réside dans le choix des constantes $c_{min} < 0$, d_{min}, $p_{max} > 0$ et d'_{min}. Il faut veiller à ce qu'une configuration où des objets sont superposés induise une énergie d'interaction positive malgré l'influence de l'énergie de connexion.

Pour cela, l'équation (2.4) impose un critère essentiel qui est de s'assurer que $p_{max} > 4|c_{min}|$.

Le choix des valeurs de d_{min} et d'_{min} est moins problématique car il n'y a pas de dépendance entre ces valeurs : il s'agit de décider d'une zone d'influence pour les énergies de connexion et de proximité.

En posant $d_{min} = \text{longueur}(o)/6$, ainsi que $d'_{min} = \text{longueur}(o)/2$, on a une zone d'influence satisfaisante. Ces valeurs sont empiriques et leur variation n'a pas une influence considérable sur le résultat qu'on obtient.

Pour éviter les superpositions, il s'est avéré nécessaire de donner à p_{max} une valeur relativement importante, sachant que l'énergie d'attache aux données est majorée par 1 en valeur absolue. Finalement, poser $p_{max} = 2$ s'est avéré donner de bons résultats et a été conservé par la suite. La partie la plus complexe est de trouver un terme de connexion qui apporte des résultats satisfaisants. La condition $p_{max} > 4|c_{min}|$ invite à essayer des valeurs de c_{min} de l'ordre de $-0,4$, mais les résultats ont rapidement fait apparaître des détections parasites (figure 2.6). Il apparaît que ce critère de connexion peut

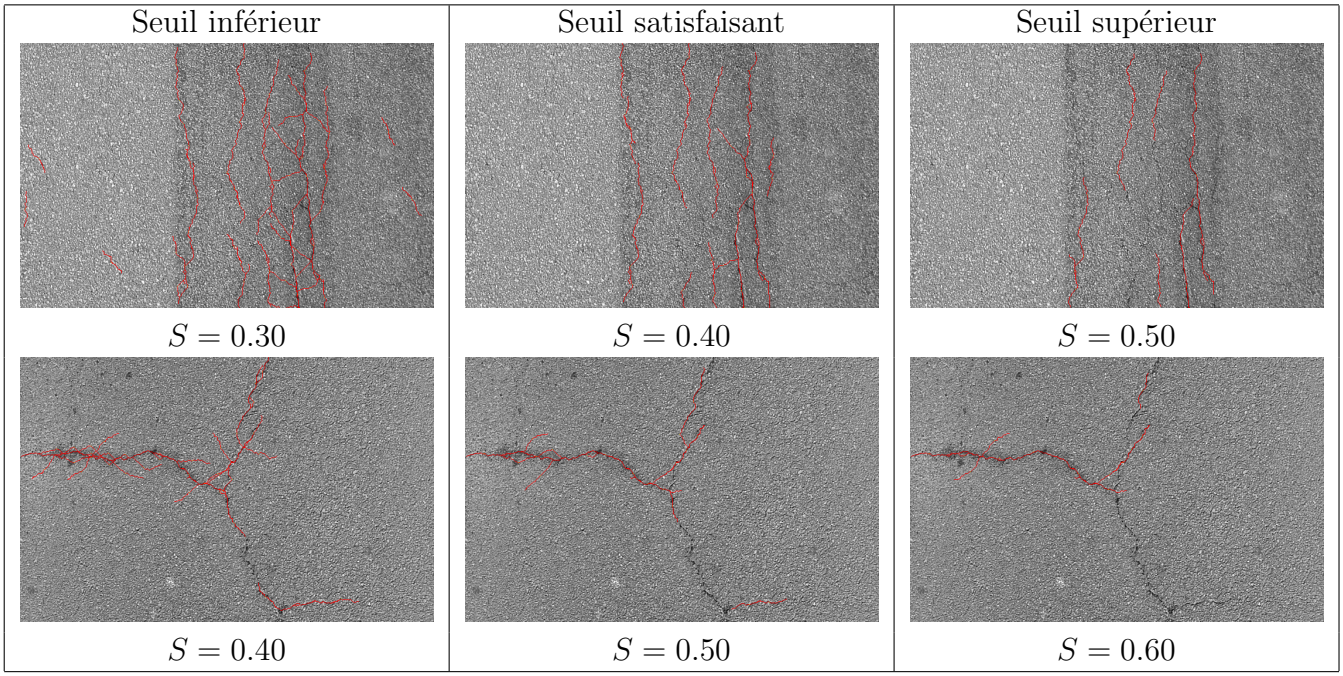


FIGURE 2.5 – Influence du seuil.

faire apparaître quelques détections parasites même pour une valeur faible relativement aux énergies d'attache aux données et de proximité, comme avec $c_{min} = -0,1$.

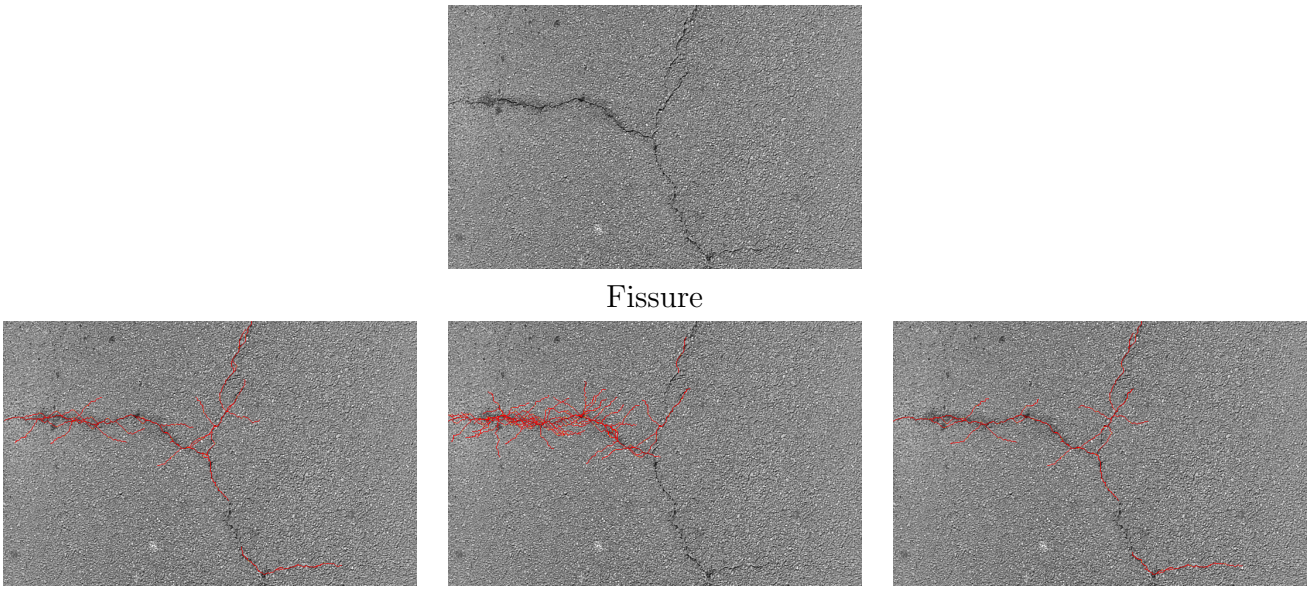
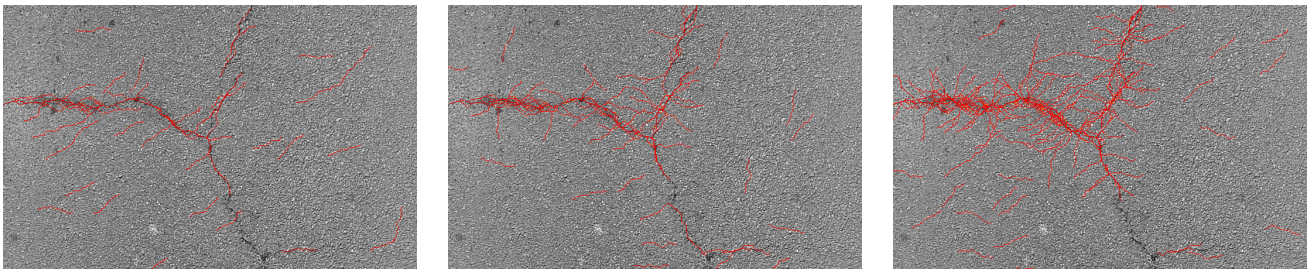


FIGURE 2.6 – Influence de c_{min} – Plus la valeur de c_{min} est importante relativement à p_{max}, plus on observe de ramifications.

2.4.3 Probabilités des noyaux

À chaque itération de l'algorithme 2.1, on choisit de réaliser une action parmi celles possibles (naissance, mort ou translation) selon une probabilité qu'il faut définir. Nous notons ces probabilités p_N, p_M et p_T. Nous avons dans un premier temps choisi de prendre des probabilités constantes, avec $p_N = 0,5$, $p_M = 0,25$ et $p_T = 0,25$. Ce choix n'a pas apporté les résultats espérés car la méthode aboutit alors à un très grand nombre de détections pour des paramètres « classiques » comme $S = 0,50$, $c_{min} = -0,1$ et $p_{max} = 2$ (figure 2.7).



2000 itérations | 5000 itérations | 10000 itérations

FIGURE 2.7 – Probabilités de noyaux constantes – $p_N = 0.5$, $p_M = 0.25$, $p_T = 0.25$.

La présence de ramifications indésirées semble causée par l'énergie de connexion, qui n'est pas bornée pour une portion $o \in x$ puisqu'elle s'écrit :

$$C(o) = \sum_{o' \in x} C(o, o') \tag{2.5}$$

Pour palier à ce problème, nous nous sommes dans un premier temps servis de l'hypothèse qu'une fissure ne recouvre qu'une faible proportion de l'image, de l'ordre de 1%, en mettant à jour les probabilités à chaque itération de manière à avoir une probabilité de naissance tendant vers 0 quand le nombre d'objets de la configuration croît. Les probabilités de mort et de translation vallent toutes deux $p_M = p_T = (1 - p_N)/2$. Cet ajustement permet d'obtenir des résultats globalement satisfaisants (figure 2.8), mais cette solution n'est pas satisfaisante car elle dénature la méthode RJMCMC qui devrait pouvoir conserver des probabilités de noyaux constantes.

2.5 Conclusion

En comparaison avec la méthode de [Ngu10], la méthode RJMCMC fait apparaître un nombre plus important de paramètres :

- T_0, la température initiale du système.
- α, coefficient de décroissance de la température du système.
- S, le seuil pour l'énergie d'attache aux données.
- γ, un paramètre de l'énergie d'attache aux données.
- c_{min}, énergie de connexion minimale entre une extrémité e de portion et une portion o'.

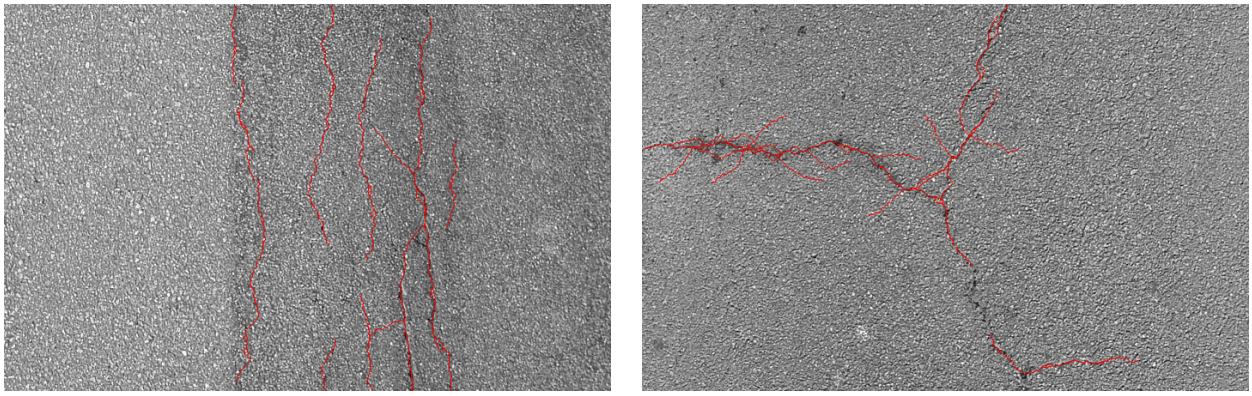


Image 1 Image 2

FIGURE 2.8 – Probabilités de noyaux variables – p_N tend vers 0 quand le nombre d'objets détectés croît. Le résultat obtenu est satisfaisant à quelques ramifications près.

- d_{min}, distance d'influence pour l'énergie de connexion.
- p_{max}, énergie de proximité maximale entre deux portions o et o'.
- d'_{min}, distance d'influence pour l'énergie de proximité.
- le pourcentage p qui détermine un nombre limite d'objets qu'on peut détecter sur la fissure.

Le calibrage des paramètres p, S, c_{min} et p_{max} est relativement délicat et le résultat trouvé dépend en grande partie de la qualité des connaissances *a priori* dont on dispose avec l'image FFA calculée.

Le principal problème que la méthode a posé est celui des ramifications de portions (figure 2.7). L'énergie de connexion d'une portion telle que nous l'avons définie est cumulable et favorise la création de réseaux de ramifications. Une solution envisagée est de ne pas prendre le cumul des énergies de connexion pour chaque portion, mais de ne prendre en considération que le minimum des énergies de connexion de chaque extrémité de portion. Ainsi, en notant x la configuration courante et $o \in x$, on obtiendrait :

$$C(o) = \underset{o' \in x}{min}(C(e_1, o')) + \underset{o' \in x}{min}(C(e_2, o')) \tag{2.6}$$

Il n'a pas été possible d'implémenter cette partie dans le cadre de ce stage faute de temps, mais les résultats attendus devraient présenter un moins grand nombre de ramifications qu'avec l'énergie de connexion actuelle. L'énergie de connexion définie en (2.6) ne favoriserait pas la création de réseaux de ramifications, et ainsi la remarque faite en section 2.4.3 serait corrigée, supprimant le paramètre p et permettant l'utilisation de paramètres de probabilité constants qui ne dénatureraient pas la méthode RJMCMC.

Chapitre 3

Analyse et comparaison des résultats

3.1 Critère de performances

Les résultats de détections se présentent sous la forme d'images binaires : les pixels de niveau 0 sont présumés être des pixels de fond, les pixels de niveau 255 sont *a priori* situés sur des fissures. Afin d'évaluer la pertinence des méthodes, nous comparons ces images calculées à des images appelées vérités de terrain, qui ont été produites manuellement. Ce sont également des images binaires et on les considère comme le résultat attendu (figure 3.1).

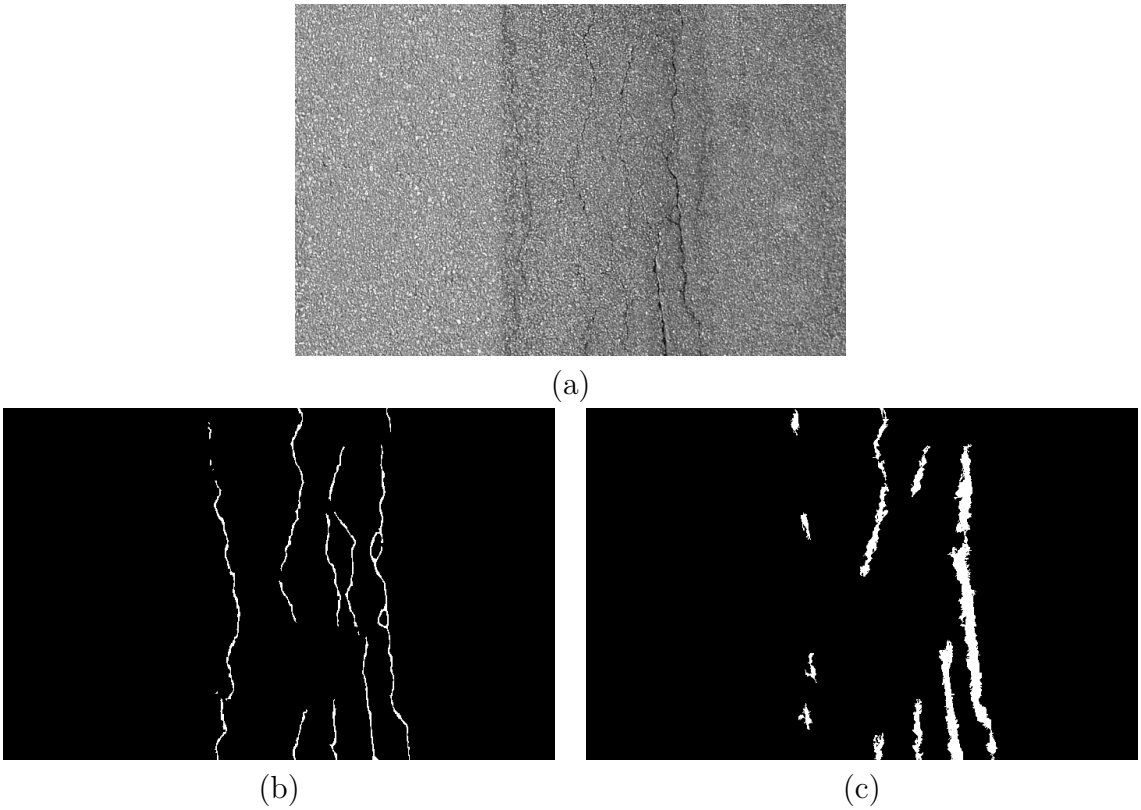
Les pixels de l'image calculée sont classées en quatre catégories (figure 3.2) :
- Les détections correctes (TP – *True Positives*).
- Les détections fausses (FP – *False Positives*).
- Les non-détections correctes (TN – *True Negatives*).
- Les non-détections fausses (FN – *False Negatives*).

Cependant, les fissures ne sont pas toujours faciles à détecter même pour l'oeil humain. En fonction de ce critère, on accorde un degré de confiance pouvant valloir entre 1 (confiance faible) et 3 (confiance forte). Plus le degré de confiance est faible, plus la zone d'acceptation pour les pixels positifs sera élargie (figure 3.3), *cf.* [CM11] pour obtenir plus de détails sur la manière d'obtenir la vérité terrain et d'établir les coefficients de confiance.

Quelques critères communs de performance sont les pourcentages de *détections correctes*, de *détections incorrectes*, ou encore de *non-détections de fissures*. Nous nous intéressons ici au critère appelé *coefficient de similarité* ou *coefficient de* Dice [CM11]. Ce critère semble être le plus significatif car il évalue le ratio entre les bonnes détections et les non-détections et il permet de synthétiser les résultats de l'ensemble des précédents critères. L'analyse des critères peut également donner une idée des points à améliorer (trop ou trop peu de détections par exemple). On note P (*Positives*) le nombre de pixels considérés comme pixels fissures dans l'image calculée.

L'expression du coefficient de similarité est la suivante :

$$\text{Dice} = \frac{2\,\text{TP}}{\text{FN} + \text{TP} + \text{P}} \tag{3.1}$$



(a) (b) (c)

FIGURE 3.1 – (a) Image de test, (b) Vérité de terrain de l'image, (c) Image binarisée calculée.

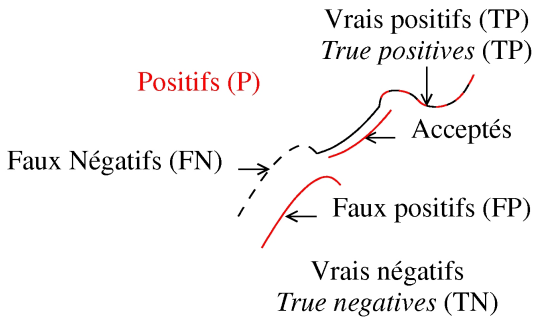


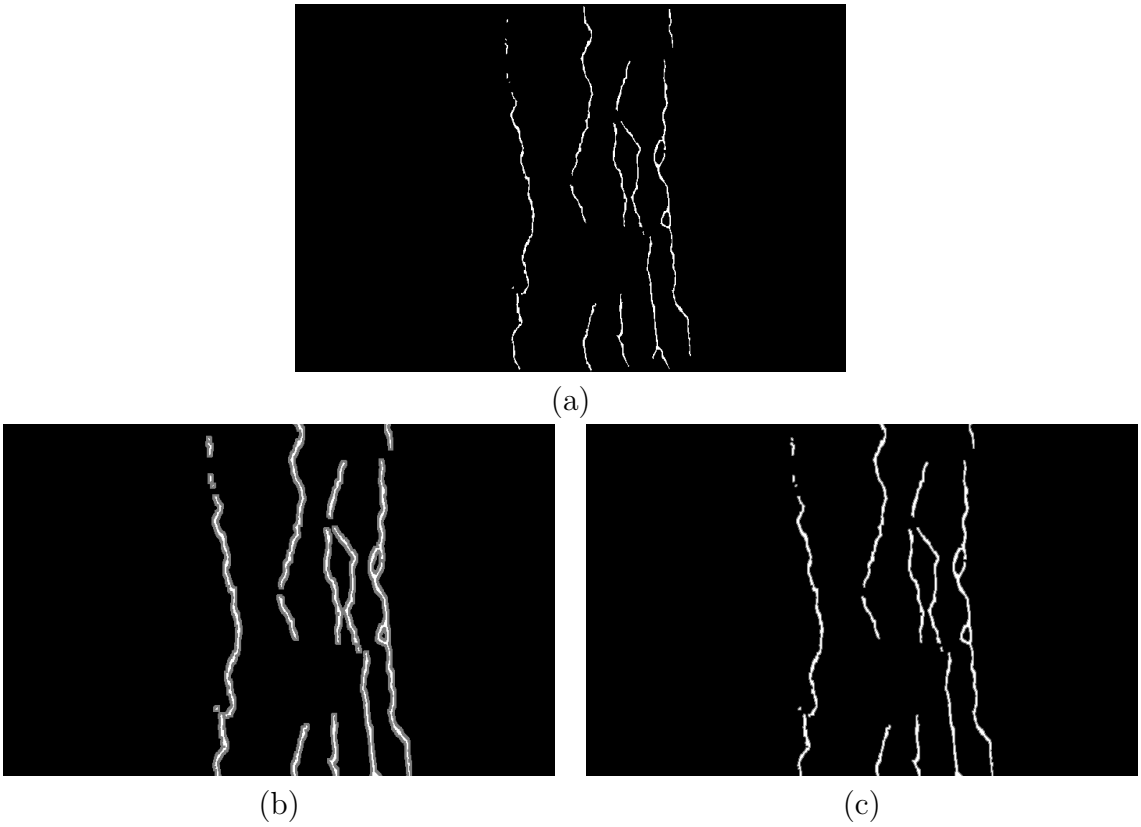
FIGURE 3.2 – Illustration des catégories de pixels.

Un critère de validation des résultats est d'avoir une valeur de Dice supérieure à $0,5$.

3.2 Résultats expérimentaux

3.2.1 Base d'images de test

Nous utilisons pour nos expériences une base de données d'images acquises de deux façons différentes : d'une part des images prises manuellement de manière statique, à l'aide d'un appareil de



(a) (b) (c)

FIGURE 3.3 – (a) Vérité de terrain, (b) Indice de confiance = 1, (c) Indice de confiance = 3.

photographie classique ; d'autre part, une base d'images capturées de façon dynamique par le véhicule utilisé au sein du réseau IFSTTAR.

FIGURE 3.4 – Système d'acquisition.

Paramètres

L'ensemble des tests effectués l'a été pour une profondeur de calcul des chemins minimums $d = 30$ et, pour la méthode RJMCMC, avec les paramètres suivants :

- $\gamma = 8$,
- $c_{min} = -0.1$,
- $d_{min} = 10$,
- $p_{max} = 3$,
- $d'_{min} = 30$,
- $p = 0.995$ (*i.e.* la probabilité de proposer une naissance devient inférieure à 1% à chaque itération où le taux de recouvrement de l'image par des objets est supérieur à p),
- Seuil S variable.

Résultats sur les images capturées de façon statique

Nous allons présenter ici des résultats de détection à partir d'images prises manuellement. La base utilisée est constituée de 5 images qui présentent des caractéristiques variables en termes de type de recouvrement, d'éclairage de la chaussée et de « clarté » de fissure.

Dans le tableau 3.5, nous répertorions pour chaque image de test la valeur de Dice calculée pour les méthodes FFA et RJMCMC avec différentes valeurs du seuil S. Les images sont situées dans l'annexe D.1.

	Méthode	FFA	RJMCMC				
image		$d = 30$	$S = 0.5$	$S = 0.6$	$S = 0.7$	$S = 0.8$	$S = 0.9$
01		0.584671	0.822363	0.70514	0.57742	0.427215	0.415866
04		0.166241	0.342702	0.214152	0.11753	0	0
12		0.381485	0.753846	0.254812	0.251607	0	0
46		0.507883	0.823529	0.59029	0.633683	0.526774	0.49837
54		0.0522147	0	0.0883352	0	0	0

FIGURE 3.5 – Tests sur des images statiques – Méthodes FFA et RJMCMC (seuils variables) et valeurs de Dice trouvées.

Ces tests mettent en évidence l'importance du choix du seuil pour la méthode RJMCMC. Les résultats sont également sensibles à la qualité de l'image FFA, qui est utilisée pour le calcul de l'énergie d'attache aux données.

Les mauvais résultats obtenus pour certaines images sont dus à certains facteurs :

- L'image 04 présente une texture particulièrement importante, ce qui nuit considérablement à la qualité de l'image FFA et rend inefficace la méthode RJMCMC.
- L'image 54 met en évidence la nuisance que cause un éclairage non uniforme, en particulier la présence d'ombres, sur la méthode FFA.
- L'image 12 obtient un résultat décevant pour la méthode FFA (Dice < 0.5), davantage dû à une mauvaise binarisation par seuillage à deux niveaux qu'à une mauvaise image FFA. On observe

cependant que l'exploitation de l'image FFA par la méthode RJMCMC présente des résultats satisfaisants.

Résultats sur les images capturées par le véhicule IFSTTAR

Nous allons présenter dans cette partie les résultats obtenus pour 6 images prises en mode dynamique par le véhicule IFSTTAR.

Dans le tableau 3.6, nous répertorions pour chaque image de test la valeur de Dice calculée pour les méthodes FFA et RJMCMC avec différentes valeurs du seuil S. Les images sont situées dans l'annexe D.2.

	Méthode	FFA	RJMCMC			
image		$d = 30$	$S = 0.4$	$S = 0.6$	$S = 0.7$	$S = 0.8$
B22296		0.142573	0	0.0883352	0	0
B22339		0.363901	0.34986	0.661033	0.691828	0.283083
C10936		0.455796	0.219347	0.669119	0.759885	0.676583
E1041		0.256024	0.23855	0.58663	0.295679	0.474603
F120		0.568199	0.530496	0.729361	0.736697	0.696948
F330		0.360587	0.475921	0.774282	0.77441	0.757792

FIGURE 3.6 – Tests sur des images dynamiques – Méthodes FFA et RJMCMC (seuils variables) et valeurs de Dice trouvées.

Ces résultats illustrent l'importance du choix du seuil pour la méthode RJMCMC. On remarque que les meilleurs résultats pour cette méthode sont obtenus avec un seuil S différent du seuil optimal pour les photographies statiques.

À l'exception de l'image B22296, les images testées satisfont toutes le critère de Dice pour la méthode RJMCMC avec un seuil approprié. Les Dice trouvés pour la méthode FFA sont en revanche peu satisfaisants car ils sont inférieurs à 0.5. Il faudrait envisager un autre système de seuillage pour l'étape de binarisation de l'image.

3.2.2 Perspectives

Le choix des paramètres des méthodes FFA et RJMCMC pourrait faire l'objet d'un apprentissage. Les images sont classifiées en fonction du type de revêtement et de leur méthode d'acquisition, et les paramètres peuvent alors être choisis selon cette classification.

Un défaut de la méthode RJMCMC présentée dans ce travail est l'absence d'information sur l'épaisseur des fissures détectées. Pour palier à ce problème, il faudrait envisager de proposer une autre nature d'objet qu'un chemin d'un pixel d'épaisseur. Il serait possible de proposer une épaisseur en attribut de chaque portion de fissure, et l'idée serait par exemple de déterminer l'épaisseur de la fissure en ce pixel.

Conclusion

Ce travail a abordé le problème de la détection automatique de fissures dans des images de chaussées en s'appuyant sur le travail proposé par [Ngu10] basé sur la *Free Form Anisothropy* (FFA), qui utilise une hypothèse de fissure relativement complète. En effet, on suppose que le chemin suivi par la fissure peut avoir une forme libre et on s'appuie sur le traitement du voisinage de chaque pixel étudié de l'image, et non un traitement ponctuel, afin de déterminer la nature du pixel. Le traitement réalisé repose sur une recherche de chemin minimum en terme de luminosité globale en chaque pixel de l'image, décomposée pour l'occasion en graphes directionnels. L'exploitation de ces données débouche sur une image FFA qui donne en chaque pixel une information sur sa probabilité d'être placé sur la fissure. Un post-traitement de l'image FFA permet d'en extraire une binarisation et de confronter les résultats trouvés à une vérité terrain afin d'évaluer la pertinence de la méthode [CM11].

Il a été nécessaire au cours de ce stage de remettre en question et discuter certains points du travail de [Ngu10] afin d'obtenir des résultats probants. En particulier, la partie concernant la binarisation de l'image FFA par seuillage à deux niveaux a été adaptée car le calcul des seuils tel qu'il était proposé par [Ngu10] aboutissait à des résultats inexploitables. Finalement, nous obtenons des résultats globalement concordants et satisfaisants au vu des critères d'acceptation présentées dans [CM11].

Nous nous sommes également intéressés à une approche supplémentaire, adaptée d'un travail proposé par [Lac04]. Cette approche Markovienne, dite *Reversible Jump Markov Chain Monte Carlo* (RJMCMC), consiste en un processus de minimisation d'une énergie comprenant un terme d'attache aux données et un terme d'interaction entre les différents objets composant la configuration. La minimisation s'effectue en itérant aléatoirement des propositions telles que des naissances, morts, translations d'objets puis en acceptant ou refusant ces propositions.

Des améliorations peuvent être apportées aux deux méthodes présentées dans ce rapport et le sujet de la détection de fissures reste ouvert aux travaux de recherche. La détection des fissures a une application directe à l'entretien des chaussées routières, mais on peut aussi envisager d'autres exploitations de ces recherches, en particulier dans le domaine médical.

D'un point de vue personnel, ce stage a été pour moi l'occasion de me familiariser avec le langage C++ et de découvrir en quoi consiste le travail de recherche en université.

Bibliographie

[BHCDZ10] Saima Ben Hadj, Florent Chatelain, Xavier Descombes et Josiane Zerubia. ■ Approche non supervisée par processus ponctuels marqués pour l'extraction d'objets à partir d'images aériennes et satellitaires ■. *Revue Française de Photogrammétrie et de Télédétection*, 194:2–15, mai 2010. Ce travail a été financé en partie par le Centre National d'Etudes Spatiales (CNES) dans le cadre du programme ORFEO.

[CM11] Sylvie Chambon et Jean-Marc Moliard. ■ Automatic Road Pavement Assessment with Image Processing: Review and Comparison ■. *International Journal of Geophysics*, (hal-00612165), Juillet 2011.

[Lac04] Caroline Lacoste. ■ *Extraction de réseaux linéiques à partir d'images satellitaires et aériennes par processus ponctuels marqués* ■. Thèse doctorale, Université de Nice-Sophia Antipolis, Septembre 2004.

[LCC01] Ping-Sung Liao, Tse-Sheng Chen et Pau-Choo Chung. ■ A Fast Algorithm for Multilevel Thresholding ■. *Journal of Information Science and Engineering 17*, pages 713–726, 2001.

[LDZ09] Caroline Lacoste, Xavier Descombes et Josiane Zerubia. ■ Unsupervised line network extraction in remote sensing using a polyline process ■. *Pattern Recognition*, 43:1631–1641, Novembre 2009.

[Ngu10] Tien Sy Nguyen. ■ *Extraction de structures fines sur des images texturées* ■. Thèse doctorale, Université d'Orléans, Novembre 2010.

[Ots79] Nobuyuki Otsu. ■ A Threshold Selection Method from Gray-Level Histograms ■. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-9(1):62–66, Janvier 1979.

Annexe A

Implémentation de [Ngu10]

A.1 Introduction

L'implémentation de l'algorithme de recherche du chemin minimum sur l'ensemble de l'image a nécessité de créer des structures adaptées au problème. Ces structures doivent permettre de :

- connaître le degré de profondeur du calcul en chaque pixel,
- connaître pour chaque pixel orienté de l'image ses listes de value et de voisin_min,
- connaître l'écart-type et le niveau de gris moyen le long du chemin issu d'un pixel pour chaque direction.

A.2 Classes adaptées

A.2.1 Classe `Sommet`

Nous avons défini une classe `Sommet` qui contient les attributs suivants :

```
1  int orientation;    //orientation liee a ce Sommet
2  int ligne;  //coordonnee en i du Sommet
3  int colonne;//coordonnee en j du Sommet
4  int k;  //nombre d'iterations executees sur ce pixel
5  int l;  //niveau de gris du pixel correspondant de l'image
6  int lmoy;   //niveau de gris moyen du chemin d'origine ce sommet
7  int sigma;  //ecart type du niveau de gris le long du chemin
8  int *sigmasum;  //somme des carres des niveaux de gris le long du chemin
9  int *value; //tableau des valuations des chemins d'origine ce sommet pour les ...
       differentes iterations
10 int *voisin_min_I;  //tableau de la coordonnee en i des voisins minimisant le ...
       chemin d'origine ce sommet pour les differentes iterations
11 int *voisin_min_J;  //tableau de la coordonnee en j des voisins minimisant le ...
       chemin d'origine ce sommet pour les differentes iterations
12 Sommet *voisins;    //ensemble des pixels constituant le voisinage de ce Sommet ...
```

```
    pour l'orientation donnee
```

Chaque objet `Sommet` est relatif à un pixel pour une des 8 orientations données (figure A.1). Par

3	5	8
2		7
1	4	6

FIGURE A.1 – Numérotation des orientations.

exemple, un `Sommet` d'orientation 1 (vers le bas-gauche) aura pour voisins les objets `Sommet` numérotés 1, 2 et 4.

Les méthodes de cette classe `findVoisinMin` et `setVoisinMin` permettent de trouver le voisin du `Sommet` de valué le plus faible et de mettre à jour les attributs du `Sommet`.

A.2.2 Classe `MatSommets`

La classe `MatSommets` définit la matrice contenant l'ensemble des sommets pour une orientation donnée.

```
1 Sommet** matrice;   //ensemble des sommets constituant le graphe...
2 int orientation;    //pour une orientation donnee
3 int rows;   //nombre de lignes
4 int cols;   //nombre de colonnes
5 int d;      //profondeur des chemins calcules
```

De cette façon, l'objet `MatSommets graphes[8]` contient l'ensemble des graphes orientés correspondants à l'image de l'étude. Les méthodes contenues par `MatSommets` permettent de :

- récupérer la matrice des Sommets constituant le graphe,
- récupérer la profondeur de calcul de chemin minimal en tout point (i, j),
- récupérer le valué en un point (i, j) de l'image pour une profondeur quelconque k.

A.3 Calcul du chemin le plus court, génération de la FFA

La fonction `calculGlobal`, faisant appel à `calculPas`, permet de calculer l'ensemble des chemins minimaux pour une profondeur et une orientation donnée sur une matrice de sommets. Après application de cette fonction pour chacune des orientations du graphe, nous exploitons le résultat grâce aux fonctions `setPiMin`, `setPiMax`, `setLmoyMin`, `setLmoyMax`, `setSigmaMin` et `setSigmaMax`, qui calculent respectivement le valué du parcours min et le valué du parcours max en chaque pixel pour les 4 directions possibles (haut-bas, gauche-droite, oblique 45° et oblique 135°) :

On rappelle que le valué en un point pour une direction donnée est égal à la somme des valués en ce point pour les deux orientations correspondantes à la direction (figure A.2).

3	4	1
2		2
1	4	3

FIGURE A.2 – Numérotation des directions.

L'image FFA est alors obtenue en faisant appel à la fonction **generateFFA**, qui a besoin pour chaque pixel de l'image de connaître les niveaux de gris moyens des parcours π_{min} et π_{fond} (*c.f.* section 1.6), ainsi que les écarts-types pour ces parcours. Cette fonction fait appel en chaque pixel à **hfun**, qui traduit le degré de cohérence entre π_{min} et π_{fond}. L'interface de **generateFFA** est la suivante :

```
1 //module "cheminMin.h"
2 double** generateFFA( char nom_image[], MatSommets ** graphes, int *** piMin, ...
      int*** orientations, int d , int* nb_l, int* nb_c);
3 //l'argument graphes est un pointeur vers le tableau des 8 graphes orientes, ...
      calcules par la fonction generateFFA
4 //l'argument piMin est un pointeur vers le tableau 2D *piMin, calcule par la ...
      fonction, contenant pour chaque pixel le value minimal parmi les 4 ...
      directions possibles (ces valeurs seront reutilisees dans le futur)
5 //orientations est un pointeur vers le tableau 2D *orientations contenant pour ...
      chaque point la valeur de l'orientation correspondant au parcours min
```

Enfin, la fonction **doubleToMat** convertit cette matrice en un objet **Mat**, avec des éléments de type **uchar** (niveaux de gris compris entre 0 et 255). Au terme de cette étape, on obtient une image FFA avec des pixels dont les niveaux de gris traduisent la probabilité d'appartenance à une fissure.

```
1 //module "cheminMin.h"
2 Mat doubleToMat(double **FFA, int nb_lig, int nb_col);
```

Exemple d'utilisation :

```
1     int d=30;                    //On choisit ici la profondeur de calcul souhaitee
2     char nom_image[]="image.pgm";  //On selectionne ici l'image de travail
3     int nb_lig, nb_col;
4     MatSommets* graphes;
5     int **piMin;
6     int **orientations;
7     double **FFA= generateFFA(nom_image, &graphes, &piMin, &orientations, d, ...
          &nb_lig, &nb_col); //Calcul de FFA[nb_lig][nb_col] pour une profondeur ...
          de chemin oriente valant d (attribue aussi leur valeur a graphes, piMin, ...
          orientations, nb_lig et nb_col)
8     Mat IFFA=doubleToMat(FFA, nb_lig, nb_col);  //Generation de IFFA, de taille ...
          nb_lig*nb_col
```

A.4 Binarisation par seuillage à deux niveaux

La solution choisie pour implémenter cette étape a été celle d'initialiser deux matrices, FFA_{bas} et FFA_{haut}, où :
- Les éléments de FFA_{bas} sont mis à 0 s'ils sont inférieurs à $seuil_B$ et intouchés sinon.
- Les éléments de FFA_{haut} sont mis à 255 s'ils sont supérieurs à $seuil_H$ et mis à 0 sinon.

Après cette initialisation, pour chaque pixel de FFA_{bas} **non nul**,
- Si le pixel correspondant dans FFA_{haut} est égal à 255, ne rien faire.
- Sinon, si l'un des 8 voisins de ce pixel est à 255 dans FFA_{haut}, mettre ce pixel à 255 dans FFA_{bas} et FFA_{haut}.

Cette opération est répétée tant qu'au moins un pixel a été mis à jour. Lorsque ce n'est plus le cas, cela signifie qu'il n'y a plus de pixel de niveau intermédiaire connecté à un pixel de niveau haut. L'algorithme prend donc fin et FFA_{haut} contient l'image FFA binarisée.

Finalement, un simple appel à la fonction suivante permet de réaliser la binarisation d'une image FFA, connaissant la profondeur d des chemins minimaux et la valeur de α souhaitée :

```
1 //module "binarisation.h"
2 void BinParSeuillage(const Mat IFFA, Mat& IBin, int d, double alpha);
```

Exemple d'utilisation :

```
1     double pourcentage=0.966;
2     Mat IBin;
3     BinParSeuillage(IFFA, IBin, d, pourcentage);
```

A.5 Connexion de composantes connexes

L'étape de connexion des composantes connexes se décompose de cette façon :
- Déterminer les contours des composantes connexes (fonction `frontiereInterieure`),
- Étiqueter les pixels par l'identifiant de leur composante,
- Trouver les pixels extrémités des composantes (fonction `isExtremite`),
- Trouver pour chaque extrémité les extrémités compatibles (même direction) d'autres composantes, tester la connexion à la composante via le chemin minimal et tester la réciprocité de la connexion.

L'interface de la fonction `ConnexionComposantes` permettant la réalisation de l'ensemble de ces opérations est la suivante :

```
1 //module "connexion.h"
```

```
2 void ConnexionComposantes(const Mat IBin, Mat& IConnexion,  MatSommets* ...
     graphes,  int** piMin, int d);
3 //IBin est la matrice de l'image FFA binarisee
4 //IConnexion est la matrice resultat post connexion
```

Exemple d'utilisation :

```
1     Mat IConnexion;
2     ConnexionComposantes(IBin, IConnexion, graphes, piMin, d);
3     //graphes permet d'acceder aux chemins minimaux calcules precedemment
4     //piMin stocke les directions des chemins minimaux en chaque pixel
5     //d est la profondeur de calcul des chemins minimaux
```

Annexe B

Implémentation de la méthode RJMCMC

B.1 Introduction

La méthode RJMCMC fait apparaître la notion d'objets entrant en interaction les uns par rapport aux autres. Il doit être possible de modifier les caractéristiques de ces objets, comme leur position sur une image, et les énergies qui leur sont attribuées doivent se mettre à jour lorsqu'un nouvel objet entre en interaction avec eux.

Ces nombreuses contraintes ont nécessité une implémentation relativement rigoureuse et la création d'une structure adaptée au problème.

B.2 Classes adaptées

B.2.1 Classes `portionFissure`, `extremite` et `idUi`

Nous avons défini une classe `portionFissure` qui contient les attributs suivants :

```
1  //Classe portionFissure
2          int id; //id de la portion de fissure
3          int i;  //abscisse centrale
4          int j;  //ordonnee centrale
5          int orientation;    //direction de la portion
6          int d;  //profondeur du chemin de la portion
7          std::vector<int> ivect; //abscisses des pixels du chemin
8          std::vector<int> jvect; //ordonnees des pixels du chemin
9          std::vector<idUi> proxvect; //vecteur des interactions de proximite
10         std::vector<idUi> connexionvect1;//vecteur d'interactions de connexion ...
               pour l'extremite 1
11         std::vector<idUi> connexionvect2;//vecteur d'interactions de connexion ...
               pour l'extremite 2
12         extremite extremite1;   //extremite 1 de la portion
13         extremite extremite2;   //extremite 2 de la portion
```

Cette classe fait apparaître la notion d'extrémité de fissure, qui est définie par la classe `extremite` définie par la position du pixel extrémité :

```
//Classe extremite
        int i;  //abscisse de l'extremite
        int j;  //ordonnee de l'extremite
```

Chaque objet `portionFissure` contient dans les vecteurs `proxvect`, `connexionvect1` et `connexionvect2` les informations sur les énergies d'interaction de proximité et de connexion relatives aux autres objets de la configuration. On utilise pour cela la classe `idUi`, qui permet d'indiquer l'identifiant de l'objet interagissant et l'énergie d'interaction résultante. La classe `idUi` possède donc les attributs suivants :

```
//Classe idUi
        long id;    //id de l'objet en interaction avec la portion
        double Ui;  //Ui entre la portion et cet objet
```

Par exemple, si une portion p_1 d'id 1 et une portion p_2 d'id 2 produisent une énergie d'interaction de proximité égale à 3, on aura :
- $p_1.proxvect = [...; < 2, 3 >; ...]$
- $p_2.proxvect = [...; < 1, 3 >; ...]$

où $< id, U_i >$ est un objet de la classe `idUi`.

B.2.2 Classe `etatImage`

La classe `etatImage` contient la configuration d'objets, qui évolue au fil des itérations, ainsi que l'ensemble des paramètres utilisés pour la méthode RJMCMC. Les attributs de cette classe sont les suivants :

```
\\Classe etatImage
        MatSommets* graphes;    //graphes (chemins minimaux)
        int** orientations;     //orientations des chemins minimaux
        double** FFA;           //image FFA
        std::vector<portionFissure> listePortions;  //liste des portions de ...
           fissures contenues dans la configuration
        double U;   //energie du systeme
        double T;   //temperature du systeme
        double gamma;//gamma d'attache aux donnees
        double S;   //Seuil d'attache aux donnees
        double alpha;//alpha de maj de la temperature apres chaque iteration (T ...
           := alpha*T)
        double cmin;//energie de connexion minimale
        double dminC;//limite de distance d'influence pour la connexion
        double pmax;//energie de proximite maximale
```

```
14          double dminP;//limite de distance d'influence pour la proximite
15          long m_id;//valeur de l'id du prochain objet cree (s'incremente a ...
                chaque naissance ou translation, meme refusee)
16          int nb_lig;//nb de lignes de l'image
17          int nb_col;//nb de colonnes de l'image
18          int affichage;  //affiche les details sur la console si affichage==1, ...
                sert aux tests unitaires
```

B.3 Exécution de la méthode RJMCMC

L'exécution de la méthode RJMCMC nécessite au préalable de choisir les paramètres à utiliser, puis de réaliser un appel à la fonction suivante :

```
1 void calculRJMCMC(MatSommets* graphes, int** orientations, double** FFA, int d, ...
      int version, cv::Mat &IRJMCMC,double T0,double alpha,double gamma,double ...
      Seuil,double cmin, double dminc, double pmax, double dminp,double ...
      pourcentage,int nb_it_max,char* src,char* dst);
2 //graphes,orientations:chemins min calcules pour la profondeur d ; FFA:valeurs ...
      FFA de l'image
3 //version=0 (version correcte) ou version=1 (variante qui calcule le ...
      recouvrement entre chaque paire de portions pour calculer les energies ...
      d'interactions)
4 //IRJMCMC est l'image qui contiendra le resultat binaire
5 //T0:temperature de depart ; alpha:coefficient de decroissance de la ...
      temperature ; gamma,Seuil:parametres d'attache aux donnees ; ...
      cmin,dminc:parametres de l'energie de connexion ; pmax,dminp:parametres de ...
      l'energie de proximite
6 //pourcentage: parametre permettant de "limiter" le nombre de portions ...
      calculees en diminuant la proba de naissance afin d'avoir approximativement ...
      un taux de recouvrement de l'image par les portions calculees egal au ...
      pourcentage.
7 //nb_it_max:nombre maximal d'iterations en cas de non convergence
8 //src:nom du fichier source (image de la fissure)
9 //dst:nom du fichier destination (portions calculees tracees dans ce fichier)
```

Exemple d'utilisation :

```
1  //graphes, orientations, FFA et d ont ete definis ou calcules au prealable
2  int version=0;
3  int nb_it_max=1500000;
4  double alpha=0.99999;
5  double T0=1;
6  double gamma=8;
7  double dminc=(double)d/3;
8  double dminp=(double)d;
9  char nom_image[]=<nom de l image>;
10 char nom_RJMCMC[]=<nom du fichier resultat>;
11
```

```
12 Mat IRJMCMC;
13 calculRJMCMC(graphes, orientations, FFA, d, version, IRJMCMC, T0, alpha, gamma, ...
       Seuil, cmin, dminc, pmax, dminp, pourcentage, nb_it_max, nom_image, nom_RJMCMC);
```

Annexe C

Exploitation des résultats

La classe `VeriteTerrain` permet le calcul des différentes données (*True Positives*, *False Negatives*, etc.) relatives à la vérité terrain d'une image et au résultat de la détection. Elle prend en compte un paramètre de confiance détaillé dans la section 3.1, ainsi que la vérité terrain et l'image de la détection calculée. Puis, le Dice s'obtient en réalisant les instructions suivantes :

```
1 //On a defini au prealable d, nom_vt (fichier contenant la verite terrain) et ...
      ICalcule est l'image calculee
2      int conf=1; //1<=conf<=3
3      VeriteTerrain VT=VeriteTerrain(nom_vt,ICalcule,conf);
4      Mat vt=VT.getVT();
5      Mat icalcule=VT.getCalcul();
6      int nTP,nTN,nFP,nFN,nPi,nPv;//nombre de True Positives, de False Positives, ...
          etc...
7      calculParametres(icalcule(Range(d,icalcule.rows-1-d),Range(d,icalcule.cols-1-d)) ...
          , vt(Range(d,icalcule.rows-1-d),Range(d,icalcule.cols-1-d)), nTP, nTN, ...
          nFP, nFN, nPi, nPv);
8      double DICE=(double)2*nTP/(nFN+nTP+nPi);
```

Annexe D

Résultats

D.1 Images statiques

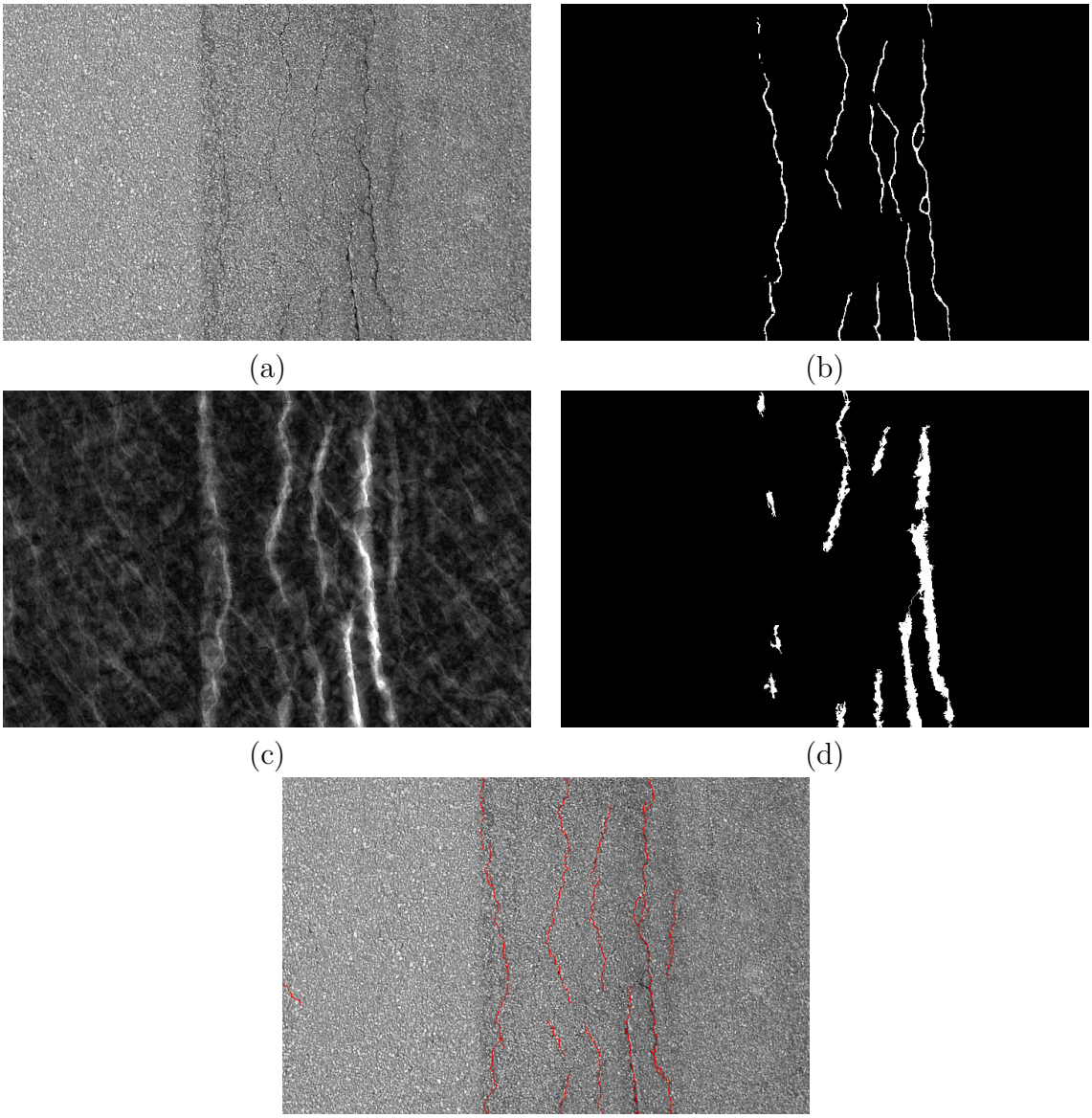


FIGURE D.1 – Image 01 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.5$).

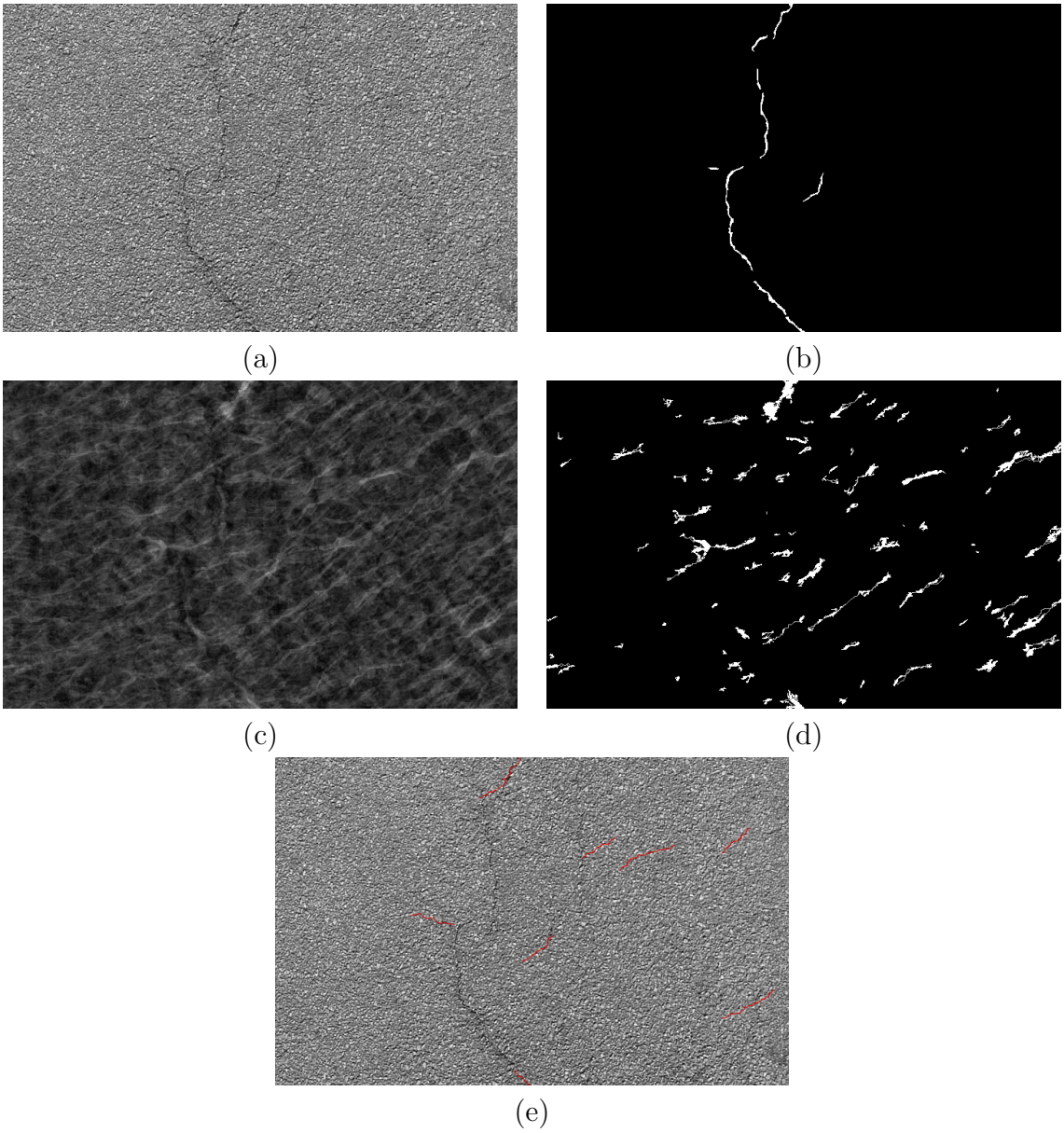


FIGURE D.2 – Image 04 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.5$).

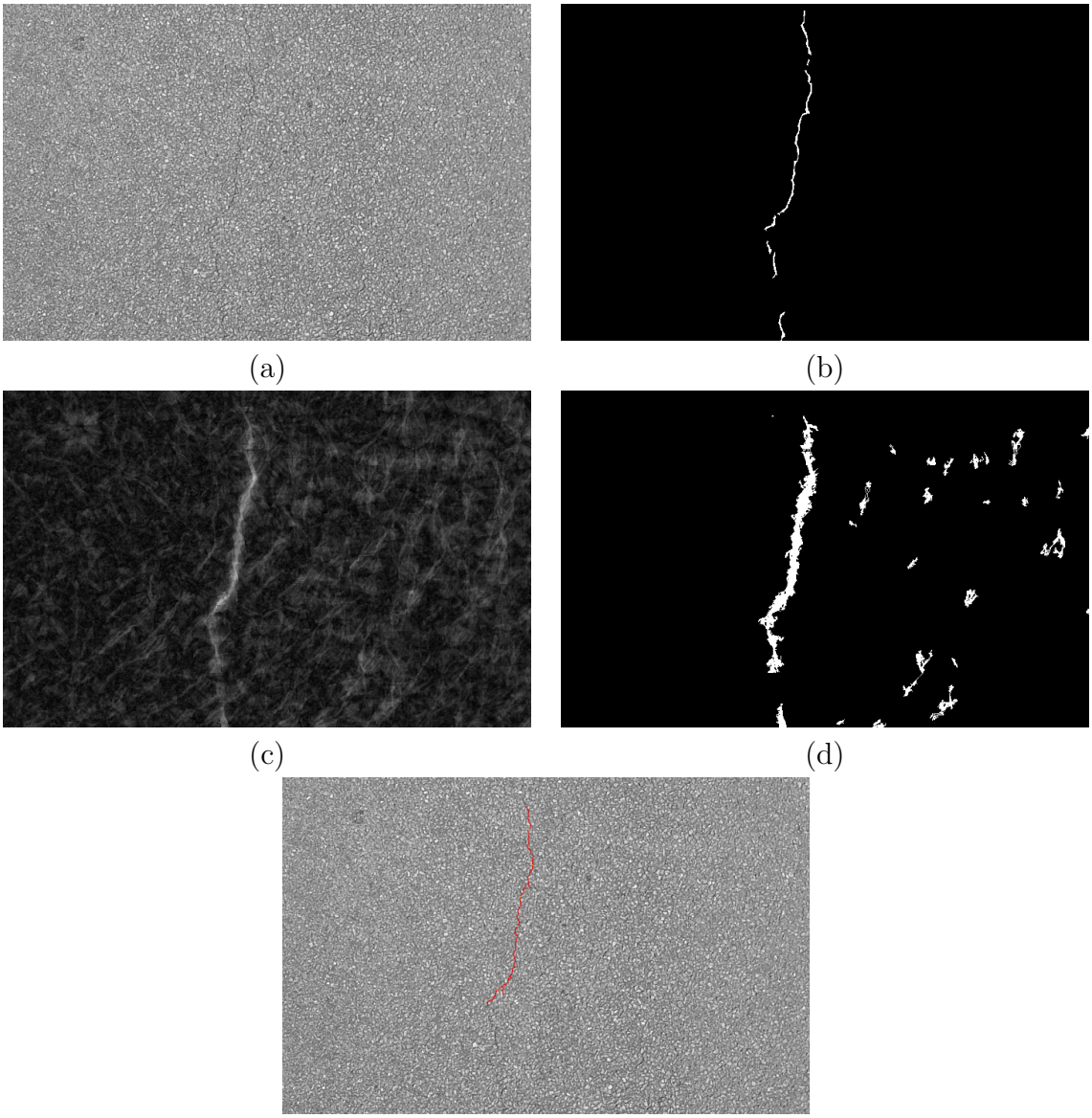


FIGURE D.3 – Image 12 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.5$).

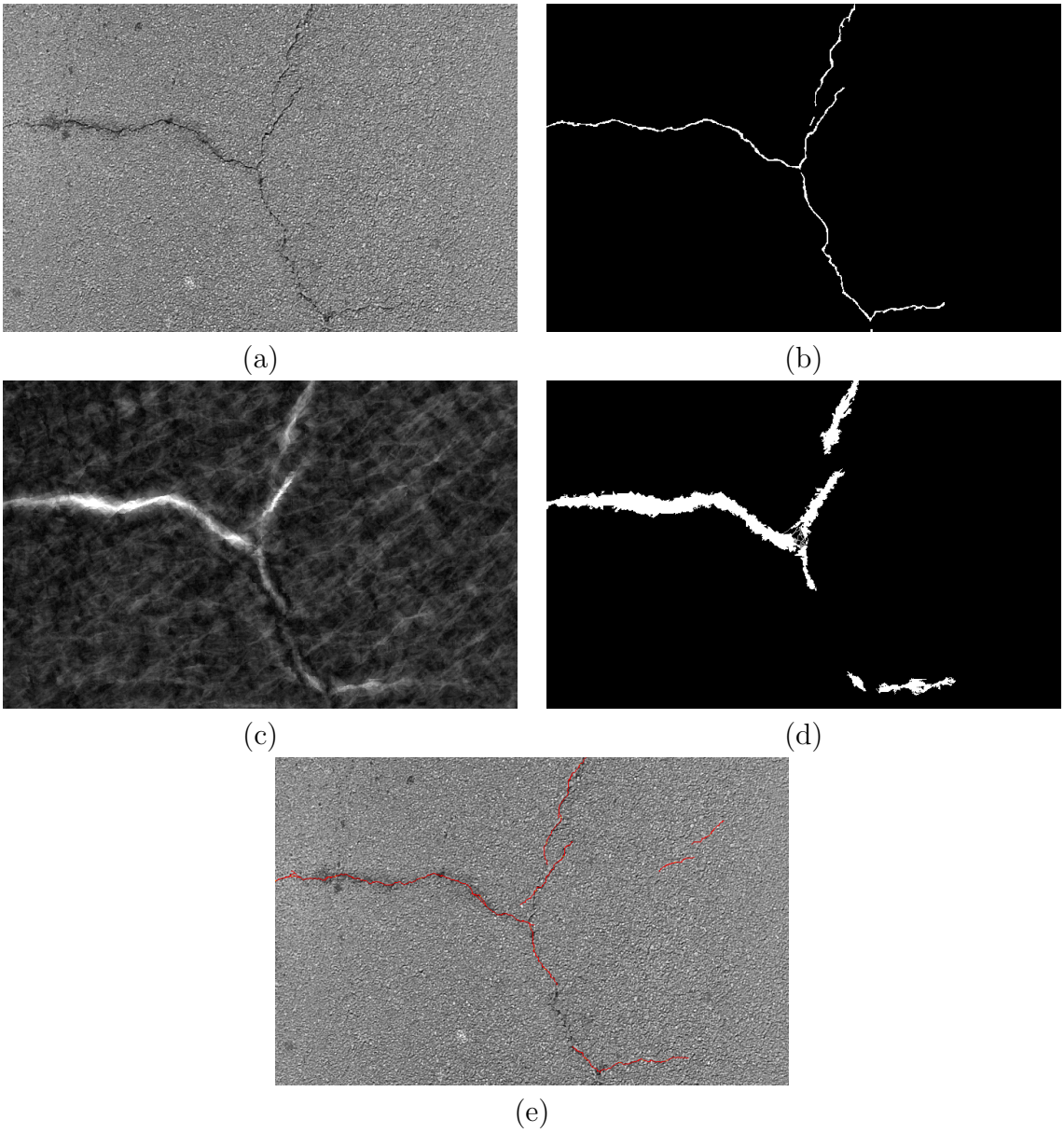


FIGURE D.4 – Image 46 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.5$).

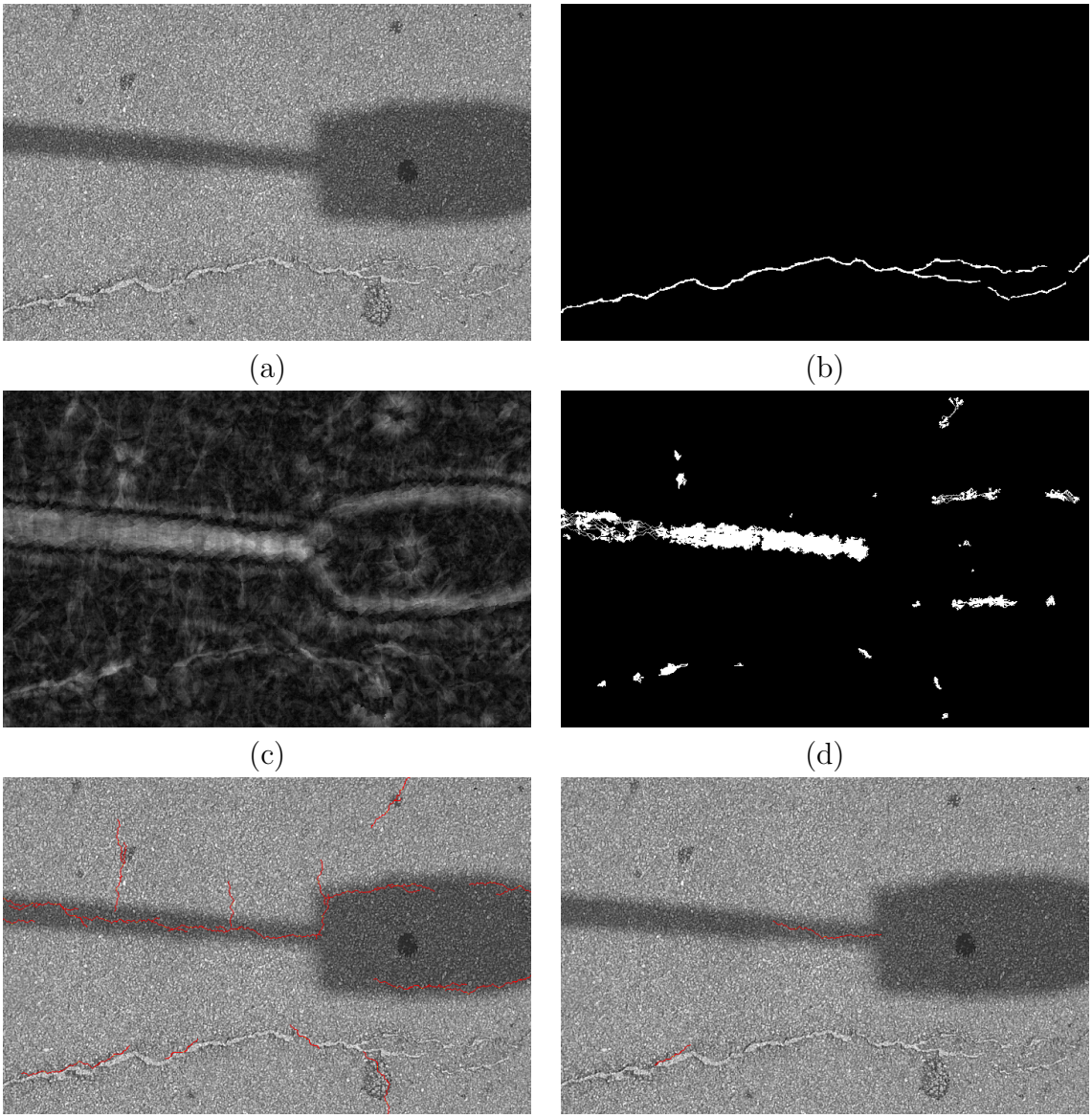


FIGURE D.5 – Image 54 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.5$), (f) Méthode RJMCMC ($S = 0.7$).

D.2 Images acquises par IFSTTAR

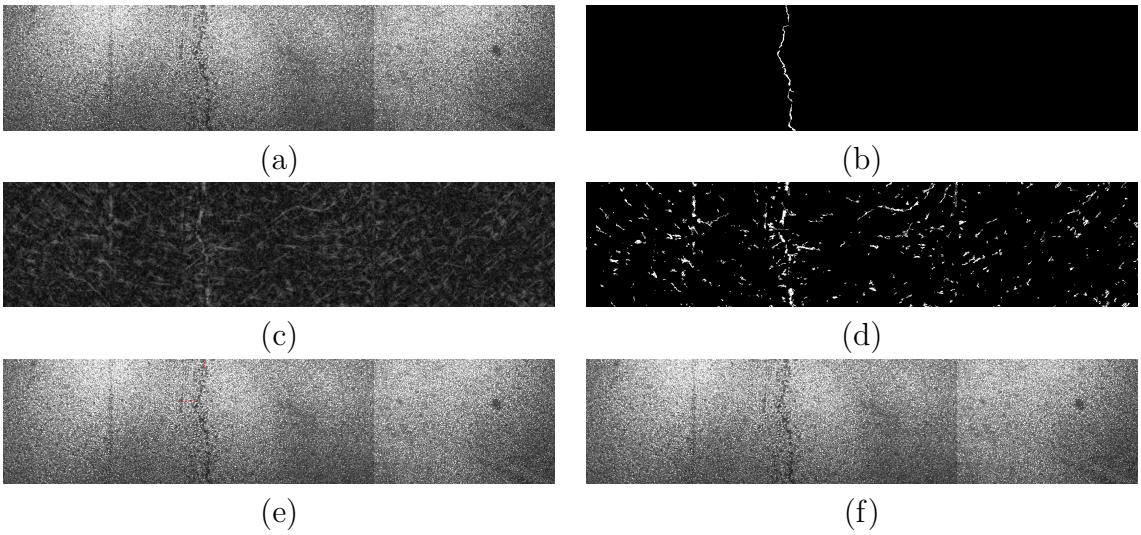


FIGURE D.6 – Image B22296 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.6$), (f) Méthode RJMCMC ($S = 0.7$).

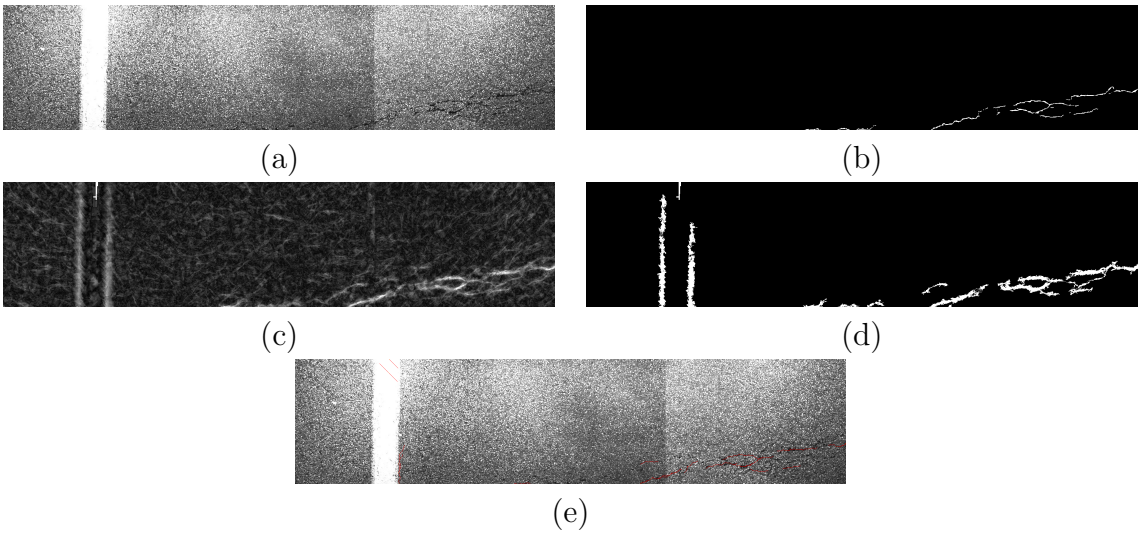


FIGURE D.7 – Image B22339 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.7$).

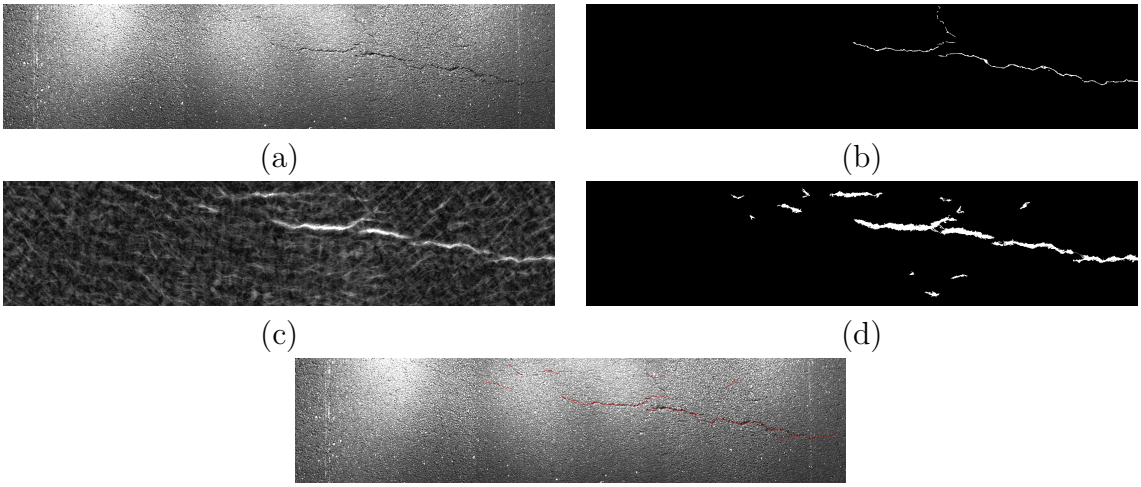


FIGURE D.8 – Image C10936 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.7$).

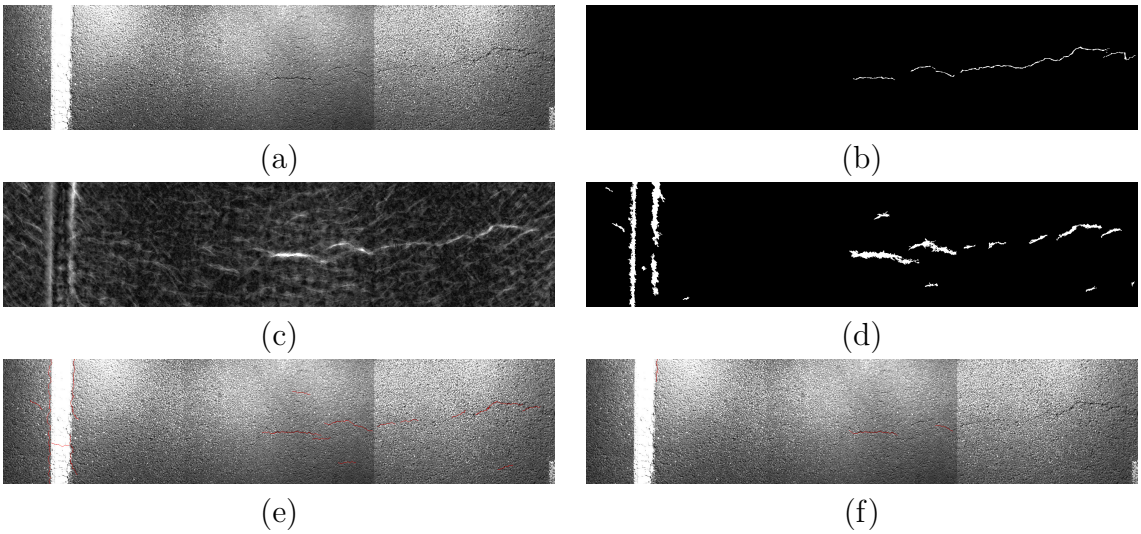
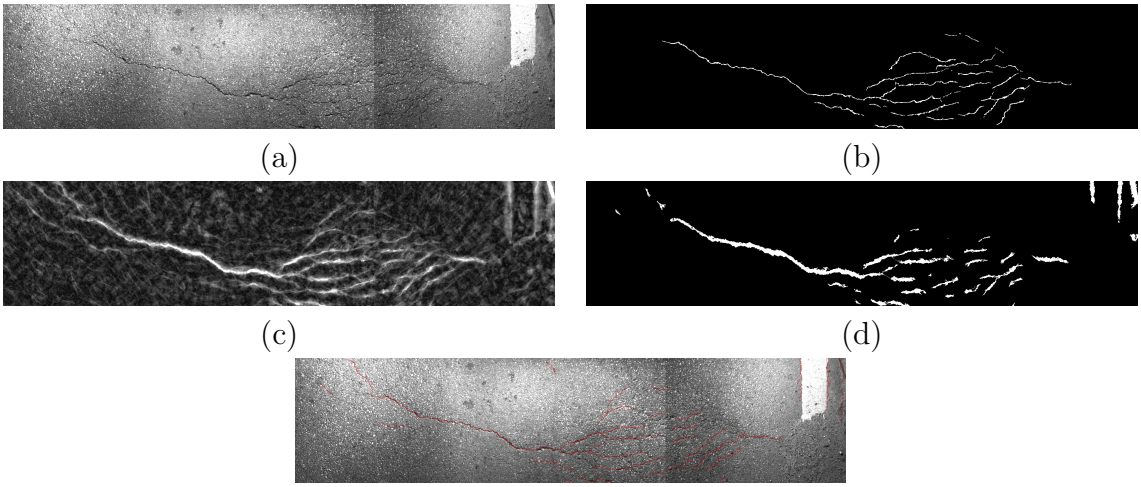
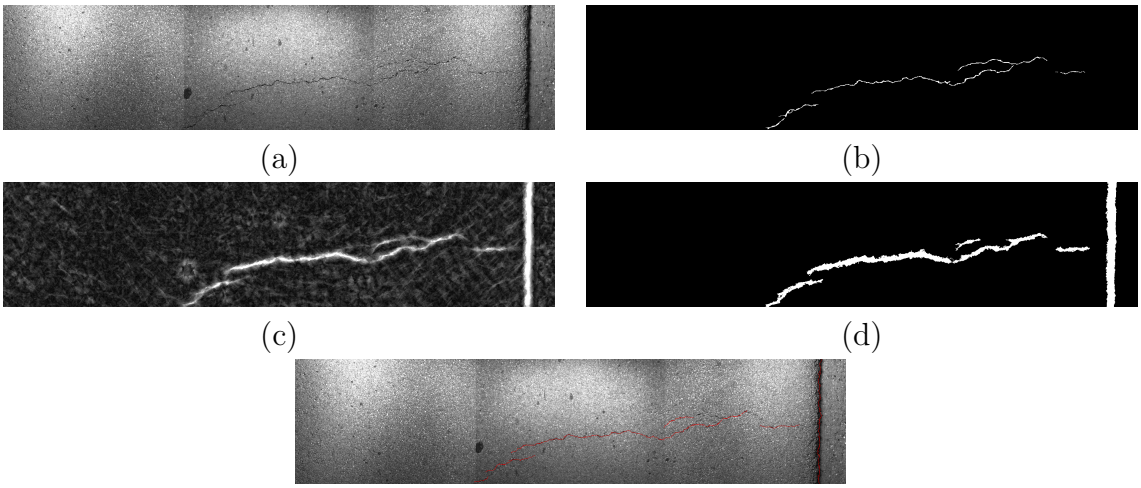


FIGURE D.9 – Image E1041 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.6$), (f) Méthode RJMCMC ($S = 0.7$).



(a) (b) (c) (d) (e)

FIGURE D.10 – Image F120 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.7$).



(a) (b) (c) (d) (e)

FIGURE D.11 – Image F330 – (a) Image de test, (b) Vérité terrain, (c) Image FFA, (d) Résultat de la binarisation, (e) Méthode RJMCMC ($S = 0.7$).