Extraction de caractéristiques sur des images acquises en contexte mobile : Application à la reconnaissance de défauts sur ouvrages d'art

Yannick Faula

N°d'ordre NNT : 2020LYSEI077

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'INSA LYON

Ecole Doctorale ED 512
(InfoMaths)

Spécialité/ discipline de doctorat : Informatique

Soutenue publiquement le 28/09/2020, par :
Yannick Faula

Extraction de caractéristiques sur des images acquises en contexte mobile

Application à la reconnaissance de défauts sur ouvrages d'art

---

Devant le jury composé de :

DEMONCEAUX, Cédric	Professeur, Université de Bourgogne	Rapporteur
NICOLAS, Henri	Professeur, Université de Bordeaux	Rapporteur
SCUTURICI, Mihaela	Maître de conférences, Université Lyon 2	Examinatrice
MEGRET, Rémi	Maître de conférences, Université de Puerto Rico (Rio Piedras)	Examinateur
CHABERT, Marie	Professeur, ENSEEIHT Toulouse	Examinatrice
BRES, Stéphane	Maître de conférences, INSA Lyon	Examinateur
EGLIN, Véronique	Professeur, INSA Lyon	Directrice de thèse

REMERCIEMENTS

Je remercie mes directeurs de thèse pour l'opportunité qui m'a été offerte au sein du LIRIS. D'abord, mon codirecteur de thèse, Stéphane Bres, car cette thèse est le fruit d'une collaboration ayant débuté durant mon Projet de Fin d'Etudes, merci pour son soutien et pour son écoute. Puis un grand merci à Véronique Eglin qui m'a encadré tout au long de cette thèse, a partagé ses idées mais également donné ses encouragements et transmis son enthousiasme.

J'adresse tous mes remerciements à Smart Aerial Machines, Stéphane Cervetti, Ganesh Kumar et Nicolas Ferrer, pour leur soutien financier, leur bienveillance, ainsi que leur accueil au sein l'entreprise.

J'exprime ma gratitude à Jean Delzers car sans sa confiance je ne serai pas au sein de sa startup.

Un grand merci à toute l'équipe IMAGINE du laboratoire ainsi que les doctorants présents à l'INSA de Lyon.

Merci également à Alain Morice dont les idées novatrices ont inspiré cette thèse.

Puis, je tiens à remercier ma mère qui avec calme et patience, à répondu à toutes mes questions quotidiennes.

Enfin, je remercie ma femme qui m'a supporté tout le long de ces années de thèse et m'aide à surmonter les moments difficiles et à me surpasser.

RÉSUMÉ

Le réseau ferroviaire français dispose d'une infrastructure de grande ampleur qui se compose de nombreux ouvrages d'art. Ces derniers subissent les dégradations du temps et du trafic et font donc l'objet d'une surveillance périodique pour détecter l'apparition de défauts. Aujourd'hui, cette inspection se fait en grande partie, visuellement par des opérateurs experts. Plusieurs entreprises testent de nouveaux vecteurs d'acquisition photo comme le drone, destinés à la surveillance des ouvrages de génie civil.

Dans cette thèse, l'objectif principal est de développer un système capable de détecter, localiser et enregistrer d'éventuels défauts de l'ouvrage. Un grand défi est de détecter des défauts sous-pixels comme les fissures en temps réel pour améliorer l'acquisition. Pour cela, une analyse par seuillage local a été conçue pour traiter de grandes images. Cette analyse permet d'extraire des points d'intérêts (Points FLASH : Fast Local Analysis by threSHolding) où une ligne droite peut se faufiler. La mise en relation intelligente de ces points permet de détecter et localiser les fissures fines. Les résultats de détection de fissures de surfaces altérées issues d'images d'ouvrages d'art démontrent de meilleures performances en temps de calcul et robustesse que les algorithmes existants.

En amont de l'étape de détection, il est également nécessaire de s'assurer que les images acquises soient de bonne qualité pour réaliser le traitement. Une mauvaise mise au point ou un flou de bougé sont à bannir. Nous avons développé une méthode réutilisant les calculs de la détection en extrayant des mesures de Local Binary Patterns (LBP) afin de vérifier la qualité en temps réel. Enfin, pour réaliser une acquisition permettant une reconstruction photogrammétrique, les images doivent avoir un recouvrement suffisant. Notre algorithme, réutilisant les points d'intérêts de la détection, permet un appariement simple entre deux images sans passer par des algorithmes de type RANSAC. Notre méthode est invariante en rotation, translation et à une certaine plage de changements d'échelle.

Après l'acquisition, sur les images de qualité optimale, il est possible d'employer des méthodes plus coûteuses en temps comme les réseaux de neurones à convolution. Ces derniers bien qu'incapables d'assurer une détection de fissures en temps réel peuvent être utilisés pour détecter certains types d'avaries. Cependant, le manque de données impose la constitution de notre propre jeu de données. A l'aide d'approches de classification indépendante (classifieurs SVM one-class), nous avons développé un système flexible capable d'évoluer dans le temps, de détecter puis de classifier les différents types de défauts. Aucun système de ce type n'apparaît dans la littérature pour la détection de défauts sur ouvrages d'art.

Les travaux réalisés sur l'extraction de caractéristiques sur des images pour la détection de défauts pourront être utiles dans d'autres applications telles que la navigation de véhicules intelligents ou le word-spotting.

Mots-clés : *Inspection d'ouvrage d'art, Points d'intérêt, Détection de lignes, Local Binary Pattern, Réseaux de neurones à convolution, Support Vector Machine*

ABSTRACT

The french railway network has a huge infrastructure which is composed of many civil engineering structure. These suffer from degradation of time and traffic and they are subject to a periodic monitoring in order to detect appearance of defects. At the moment, this inspection is mainly done visually by monitoring operators. Several companies test new vectors of photo acquisition like the drone, designed for civil engineering monitoring.

In this thesis, the main goal is to develop a system able to detect, localize and save potential defects of the infrastructure. A huge issue is to detect sub-pixel defects like cracks in real time for improving the acquisition. For this task, a local analysis by thresholding is designed for treating large images. Ths analysis can extract some points of interest (FLASH points : Fast Local Analysis by threSHolding) where a straight line can sneak in. The smart spatial relationship of these points allows to detect and localise fine cracks. The results of the crack detection on concrete degraded surfaces coming from of images of infrastructure show better performances in time and robustness than the state-of-art algorithms.

Before the detection step, we have to ensure the acquired images have a sufficient quality to make the process. A bad focus or a movement blur are prohibited. We developed a method reusing the preceding computations to assess the quality in real time by extracting Local Binary Pattern (LBP) values. Then, in order to make an acquisition for photogrametric reconstruction, images have to get a sufficient overlapping. Our algorithm, reusing points of interest of the detection, can make a simple matching between two images without using algorithms as type RANSAC. Our method has invariance in rotation, translation and scale range.

After the acquisition, with images with optimal quality, it is possible to exploit methods more expensive in time like convolution neural networks. These are not able to detect cracks in real time but can detect other kinds of damages. However, the lack of data require the constitution of our database. With approaches of independant classification (classifier SVM one-class), we developed a dynamic system able to evolve in time, detect and then classify the different kind of damages. No system like ours appears in the litterature for the defect detection on civil engineering structure.

The implemented works on feature extraction on images for damage detection will be used in other applications as smart vehicle navigation or word spotting.

Keywords : *Structural Health Monitoring, Point of interest, Line Detection, Local Binary Pattern, Convolution Neural Network, Support Vector Machine*

Table des matières

Table des figures

Liste des tableaux

1

Contexte et motivations

1.1 Introduction

L'interaction avec le monde extérieur passe en grande partie d'abord par le visuel. La perception visuelle humaine permet d'effectuer une tâche ; un travail sous des conditions diverses et en un temps limité. La capacité de vision permet de constater quelque chose, de l'analyser et par la suite peut déboucher sur une prise de décision. Ces informations sont captées dans des environnements divers (intérieur et/ou extérieur) par des capteurs optiques, par exemple un appareil photo numérique. Les tâches récurrentes demandent du temps à un homme et peuvent également nécessiter des moyens lourds. C'est pourquoi, l'automatisation de certaines tâches répétitives permet un gain de temps tout en ayant une qualité de résultat comparable. La phase d'automatisation n'a pas vocation à remplacer l'homme mais vient l'épauler dans des tâches récurrentes, consommatrices en temps ou contraignantes physiquement.

On définit un vecteur comme étant un moyen de transmettre de l'information. Dans un processus d'automatisation des tâches, nous devons utiliser un vecteur qui va acquérir et/ou traiter de l'information. Un vecteur peut être un robot terrestre, aérien ou naval. Il peut s'agir d'une voiture, d'un bateau, d'un sous-marin, d'un drone aérien (ou UAV :Unmanned Aerial Vehicule). Tous ces systèmes sont mobiles dans leur environnement et peuvent être guidés par l'homme ou être en totale autonomie. De la même façon qu'un homme se déplace dans des environnements inconnus, il faut pouvoir donner la capacité de déplacement automatique au vecteur. Un vecteur d'acquisition qui suit une trajectoire quelconque doit pouvoir répondre de la même manière, quels que soient les

FIGURE 1.1 – Exemple de vecteur d'acquisition : le drone aérien (UAV)

obstacles et difficultés rencontrés. Pour cela, on va doter le vecteur d'une capacité de calcul et définir les actions qu'il sera capable de réaliser. Le vecteur d'acquisition et la capacité de calcul constituent un robot autonome. Pour qu'une tâche soit bien effectuée par un robot autonome, deux conditions principales sont à remplir :

— l'acquisition doit être réalisée avec le plus de précision possible. On veut éviter de re-acquérir des données. Cela implique qu'il y ait assez de données mais aussi que les données récoltées soient de suffisamment bonne qualité pour achever une tâche. De plus, des traitements automatiques peuvent être menés afin de réaliser au mieux l'acquisition.

— l'analyse des données doit être assez déterministe et le système doit répondre selon l'expertise du domaine d'application. Le traitement peut se faire pendant ou après l'acquisition. Dans certains cas, l'analyse est nécessaire en temps réel pour pouvoir effectuer une action précise (un mouvement, une décision etc) par la suite. En cas d'aléa, le vecteur autonome doit pouvoir réagir en conséquence pour répondre aux attentes.

Certains domaines d'applications nécessitent un niveau d'autonomie complet. Dans le cadre de cette thèse, les données seront uniquement acquises à l'aide d'appareils photos mobiles. Mais, d'autres capteurs pourraient être utilisés comme les caméras thermiques, les LIDARs, etc. Et pour augmenter encore le niveau d'autonomie (au sens prise de décision), plusieurs algorithmes peuvent être exploités comme la planification, la localisation, la modélisation 3D, la détection, la reconnaissance d'objets, le suivi en temps réel.

Dans ce travail de thèse, nous retrouverons à de nombreuses reprises, les notions de détection, localisation, classification et reconnaissance des objets. Le plus souvent, les notions de détection et de localisation seront confondues. Cependant, nous pouvons tout de même les dissocier. La détection consiste à déterminer la présence d'un objet, la localisation consiste à situer un objet dans l'espace et/ou dans le temps. La classification et la reconnaissance constitueront les étapes de catégorisation d'un objet par rapport à des caractéristiques similaires à un modèle et un environnement. Cependant, nous observons dans la littérature que la phase de détection comprend généralement une étape

FIGURE 1.2 – Exemple d'inspection des ouvrages d'art (source internet)

de localisation d'où l'assimilation d'une notion avec l'autre. Dans le cas contraire, nous préciserons l'action réalisée dans les technologies testées pour parvenir à une localisation lorsque celle-ci est disjointe de la phase de détection.

Diverses applications d'analyse d'image ou de séquences d'images acquises en contexte mobile sont présentées dans les sections 1.3 et dans le travail de thèse réalisé, nous nous sommes focalisés sur l'inspection des ouvrages d'art dont une description fine est apportée dans la section suivante 1.2.

1.2 Contexte industriel : Inspection des ouvrages d'art

La SNCF dispose d'un réseau de grande ampleur composé de milliers d'ouvrages d'art, et de milliers de kilomètres de rails. Ce réseau doit respecter des exigences minimales qualifiées de base. Par exemple, il s'agit d'assurer la résistance mécanique et la stabilité des ouvrages d'art, ou encore de maintenir la sécurité d'utilisation et l'accessibilité. Ce réseau doit alors résister au trafic important des trains mais aussi aux intem-

FIGURE 1.3 – Exemples d'avaries

péries naturelles telles que la pluie, le vent, les températures très basses ou très élevées. Il subit donc des dégradations importantes et fait aussi l'objet d'une surveillance régulière et périodique. Aussi, des défauts de construction peuvent altérer plus rapidement les ouvrages d'art dès leur conception. Cette surveillance se fait actuellement avec des opérateurs qui effectuent une inspection visuelle à l'oeil nu et par conséquent de manière non exhaustive. Le coût de cette surveillance est élevé et nécessite des moyens lourds comme l'illustre la figure 1.2. On comprend donc l'intérêt que pourra apporter l'automatisation du processus d'inspection. Par exemple, un ouvrage d'art de 600 mètres de long peut représenter une dizaine de milliers de photos.

L'inspection des ouvrages d'art au centre de cette thèse pourra se retrouver désignée également par le mot-clé de *Structural Health Monitoring* (SHM) dans la littérature.

L'entreprise Smart Aerial Machines, dans le cadre d'un partenariat avec la SNCF, a eu pour objectif de développer un drone capable d'inspecter les ouvrages d'art. L'entreprise a mis à disposition les moyens pour réaliser cette thèse dans de bonnes conditions et m'a fourni des données pour développer les algorithmes proposés.

(a) (b) (c)

FIGURE 1.4 – Exemples de défauts. (a) Défaut linéaire (large fissure verticale) (b) Défaut surfacique présentant les mêmes caractéristiques visuelles qu'un défaut linéaire (faïençage) (c) Défaut ponctuel (épaufrure)

1.2.1 Type de défauts

Lors d'une inspection, l'objectif d'un expert est de comprendre où se situent les détériorations et de comprendre comment cela impacte la structure globale d'un ouvrage d'art. La connaissance des forces s'exerçant sur l'ouvrage est capitale pour assurer sa bonne maintenance et sa pérennité. La diversité des avaries impacte ainsi de manière diverse la structure en l'affectant de façon plus ou moins importante. Ces avaries sont généralement catégorisées en 3 grands ensembles :

— **défauts linéaires** : ce sont principalement les fissures ou les fractures se présentant sous forme d'un tracé continu marquant une discontinuité sur une surface.

— **défauts surfaciques** : ils se présentent sur une zone de l'infrastructure qui peut se décrire par sa superficie. On retrouve les zones d'humidités, le faïençage, les éclatements de béton, les décollements d'enduit ou les lacunes de bétonnage.

— **défauts ponctuels** : ce sont des éclats accidentels impactant l'ouvrage sur une zone isolée comme les arêtes. Il peut s'agir d'un éclatement de béton ou d'une épaufrure.

Dans toutes ces catégories, certains défauts peuvent avoir des caractéristiques similaires mais sont répertoriés de manière différente car ils peuvent nécessiter des moyens différents pour l'entretien courant et/ou la réparation.

1.2.2 Objectifs

Dans le cadre de ce projet avec la SNCF, plusieurs types d'ouvrages d'art, se distinguant par leur matériaux et méthodes de construction, sont ciblés mais uniquement ceux en béton seront analysés dans ce travail de thèse. Les avaries présentes sur ces ouvrages sont plus difficiles à localiser car on veut les identifier le plus tôt possible (si possible avant même que ce soit visible à l'oeil nu). Le cahier des charges au départ liés à ce type de structure était le suivant :

— Elaborer un drone capable d'effectuer des traitements embarqués.

— Acquérir des images de bonne qualité. La qualité de l'image doit être suffisante pour réaliser des traitements et une reconnaissance de défauts très peu perceptibles.

— Détecter automatiquement les avaries. Une attention particulière sera accordée à la reconnaissance de fissures.

— Analyser les photos ainsi acquises pourra dans le meilleur des cas, permettre la réalisation d'un modèle 3D de l'ouvrage.

Plusieurs objectifs intermédiaires ont fait leur apparition tout au long de la thèse. Notamment, la détection des ombres dans des images, la détermination de la qualité d'une image, ou encore l'identification d'un défaut présent sur plusieurs images.

1.2.3 Contraintes

La surveillance des ouvrages d'art par drone implique un traitement des informations dans un système embarqué. Les avaries sont également très diverses. Pour qu'une inspection soit considérée comme pertinente pour les experts en inspection, il faut pouvoir détecter des fissures de l'ordre du dixième de millimètre. Pour cela, la résolution des images acquises doit être suffisante pour permettre cette reconnaissance.

Compte tenu des contraintes imposées, on admettra que la résolution minimale pour détecter une fissure est de 0.5 mm/pixel. Il s'agit d'une valeur limite d'après les expérimentations réalisées par Smart Aerial Machines. Avec une moins bonne résolu-

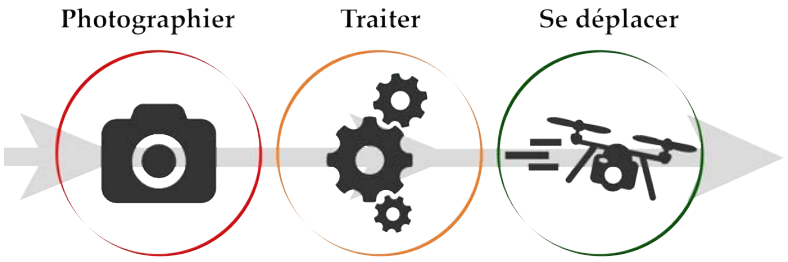


FIGURE 1.5 – Processus

tion, on ne peut pas espérer détecter les fissures les plus fines qui sont recherchées. On pourrait envisager une résolution plus importante, mais cela réduirait la surface inspectée sur chaque image et multiplierait aussi le nombre de prises de vue à réaliser et à analyser.

1.2.4 Présentation du processus

Comme le montre la figure 1.5, le processus se décompose en 3 phases principales s'effectuant de manière itérative.

Photographier - le vecteur d'acquisition sera lancé pour prendre des photographies. Ces dernières sont prises automatiquement dès que les conditions optimales du vecteur sont atteintes. Par exemple, une technologie brevetée par Smart Aerial Machines permet de positionner l'axe optique de l'appareil photo de manière orthogonale à la surface. La gestion de la distance à cette surface se fait également automatiquement.

Traiter - Les images seront traitées et une décision sera alors prise. Le traitement inclut par exemple la détection de fissures, mais aussi tout ce qui concerne la vérification de la qualité de la photo. Dans ce processus, on se limite uniquement à la phase de détection. On ne cherche pas à calculer la largeur d'une fissure. On pourra également inclure des algorithmes d'appariement entre les images précédentes et l'image en cours. De plus, la reconstitution d'une mosaïque d'images peut permettre de recaler la position du drone par rapport à un modèle 3D existant.

Se déplacer - Par la suite, le vecteur pourra se déplacer pour 2 raisons principales ; soit

dans le but de reprendre une photo avec une plus grande résolution pour permettre la caractérisation fine d'une fissure, dans le cas d'une détection par l'algorithme embarqué, soit se déplacer et se préparer à la prochaine acquisition photo. Dans ce dernier cas, le déplacement sera fait de façon à ce que l'on ait un recouvrement minimum nécessaire entre les images. Cela se détermine à l'aide de capteurs supplémentaires. Un recouvrement suffisant permet d'assurer l'inspection exhaustive de la surface mais aussi les appariements d'images.

Le travail de thèse se concentre donc uniquement sur la partie traitement des photographies quel que soit le vecteur d'acquisition utilisé. C'est pourquoi, les contributions apportées peuvent être utiles dans d'autres domaines d'application présentés dans la section suivante.

1.3 Autres domaines d'application des travaux de thèse

1.3.1 Mobilité

Automobile- Les voitures autonomes impliquent un grand nombre de capteurs notamment de caméras numériques. Ces dernières fournissent des informations qui aident à la conduite ou apportent un appui pour la sécurité du conducteur. Dans ce contexte, l'information doit être traitée en temps réel et doit être la plus juste possible.

Robot- Les grands de l'industrie, comme Amazon, travaillent sur des systèmes de livraisons autonomes. Des drones ou des robots ont recours aux caméras pour pouvoir se déplacer, éviter des obstacles et prendre un objet et le livrer. L'interaction avec un environnement très ouvert est donc primordiale et les aléas sont à gérer.

1.3.2 SLAM et odométrie

Le domaine de la photogrammétrie et du SLAM (Simultaneous Localization and Mapping) requiert une attention particulière pour une acquisition de bonne qualité. Le traitement étant généralement réalisé en différé, il faut s'assurer que les images acquises

soient de qualité suffisante. Le SLAM est la faculté d'un vecteur à savoir se situer et à mémoriser l'espace déjà visité en reconstruisant en 3 dimensions son environnement avec une erreur assez variable. Dans ce domaine, la modélisation 3D de bâtiments, de monuments historiques etc. aide ensuite les ingénieurs dans leur expertise.

1.4 Contributions

Les contributions de la thèse sont diverses mais elles s'articulent autour du même processus général présenté ci-dessus. Nous pouvons donc lister des contributions dans ces domaines :

— **Détection des fissures temps réel :** Un algorithme quasi temps réel de détection des fissures a été développé, désigné par l'acronyme FLASH (*Fast Local Analysis by ThresSHolding*), et avec lui, des solutions de traitement de données optimisées pour du temps réel. La technique proposée est bas-niveau et peut servir dans beaucoup d'autres applications, notamment dans les autres contributions dépassant le cadre de la détection d'avaries sur les ouvrages d'art.

— **Appariement d'images :** Un nouvel algorithme permettant d'apparier les images est proposé. Il permet de réutiliser les calculs faits lors de la détection de fissures pour gagner en temps de traitement. Il est tout particulièrement adapté au type d'images que nous avons à traiter, et pour lesquelles les algorithmes plus classiques ne donnent pas de très bons résultats, pour des images très peu contrastées et très peu texturées.

— **Détection de flou temps réel :** Un algorithme de détection de flou temps réel, couplé et basé sur les extractions de l'algorithme FLASH, a été élaboré. Ces deux algorithmes sont conjointement utilisés pour la détection des fissures. Il s'agit ici de déterminer si une image est de qualité suffisante pour pouvoir effectuer les tâches prévues (par exemple la détection de fissures ou la modélisation 3D).

— **Détection de défauts surfaciques en temps différé :** Des algorithmes de détection de défauts surfaciques, tels que les éclatements de béton, sont aussi proposés. Ils se basent sur l'utilisation des réseaux de neurones et des SVM (Support

Vectors Machines). Il n'est plus question de la contrainte de temps pour cette partie de l'analyse, car elle n'est pas destinée à être embarquée dans le drône et réalisée en temps contraint. Il s'agit plutôt d'une analyse "off line", une fois que l'ensemble des images a été acquis. Précisons que ce n'est pas le cas de la détection des fissures qui doit se faire au moment de l'acquisiton des images, donc in situ, car cela déclenche une acquisition supplémentaire de qualité supérieure sur laquelle se basera l'analyse des caractéristiques de la fissure en question.

— **Base d'images annotées :** Une nouvelle base de données de fissures et défauts surfaciques qu'il nous a été nécessaire de constituer pour pouvoir tester les nouveaux algorithmes proposés. Par ailleurs, un outil simple de génération de fissures, ainsi qu'un outil de labellisation viennent s'ajouter à cette contribution.

1.5 Organisation du mémoire

Dans sa structuration, les chapitres 2, 3, 4 sont consacrés aux techniques d'extraction de vecteurs de représentation et leur manipulation. Ils présentent essentiellement des techniques bas-niveau, qui restent les plus utilisées pour la détection temps réel de structures fines. Le chapitre 5 introduit des nouvelles techniques d'apprentissage automatique apportant des connaissances plus fines et plus robustes pour la détection de défauts surfaciques.

Plus en détails, le chapitre 2 présente notre première contribution FLASH utilisée dans la détection de défauts de type linéaires (principalement les fissures). Le chapitre 3 décrit la détermination de la qualité d'une image acquise. Dans le chapitre 4, nous introduisons notre algorithme d'appariement d'images utilisant FLASH. Puis le chapitre 5 présente la détection des défauts surfaciques à l'aide de techniques d'apprentissage. Quelques autres applications et contributions sont présentées dans le chapitre 6 permettant d'apprécier l'ouverture à d'autres domaines. Enfin, nous concluons par le chapitre 7 et par des perspectives dans le domaine de la surveillance automatisée des ouvrages d'art.

2

Détection de défauts : les fissures

FIGURE 2.1 – Exemples de fissures de différentes tailles sur divers murs en béton.

2.1 Introduction

L'extraction de défauts linéaires est la partie la plus importante dans la reconnaissance de défauts sur ouvrages d'art. Elle permet de connaître les zones d'un ouvrage où le travail mécanique a lieu et de constater très tôt les zones sensibles de l'ouvrage. Une fissure peut également contribuer à l'apparition de plusieurs autres types de défauts comme des infiltrations d'eau, ou des éclatements de béton. Connaître la présence et l'emplacement des fissures est donc important. On définira une fissure de la façon suivante :

Une fissure est une fente, une faille linéaire au tracé plus ou moins régulier. Ses caractéristiques sont sa longueur, son orientation par rapport à l'ouvrage, et son ouverture.

La fissure peut présenter des caractéristiques visuelles et s'apparenter à un contraste local très différent de son voisinage. Ainsi, les techniques de détection de contours sont utilisées dans la littérature. Des exemples de fissures sont présentés sur la figure 2.1. La détection des fissures est utile pour l'inspection des structures en béton du réseau SNCF, mais aussi dans beaucoup d'autres domaines de contrôle qualité comme l'analyse des matériaux ou l'inspection des infrastructures routières.

2.2 Etat de l'art

Une définition de fissure est présentée dans [Chambon10]. Elle décrit ses caractéristiques visuelles dans une image :
Une fissure est un ensemble de pixels avec un niveau de gris plus foncé que le fond de l'image. De plus, une fissure peut être vue comme un ensemble de petits segments avec des orientations différentes et connectés entre eux. On distingue donc deux caractéristiques, le contraste local et la cohérence spatiale selon l'orientation de la fissure, sur lesquelles les algorithmes vont se baser pour effectuer la détection. Dans la thèse de [Nguyen10], 4 étapes ont été identifiées : prétraitement, segmentation, post-traitement et extraction des attributs, classification. Elles sont souvent menées séquentiellement selon un pipeline traitant les images ; soit selon leur contenu en niveau de gris ou couleur (méthodes variationnelles), soit selon leur contenu fréquentiel, soit selon une projection des informations de l'image dans des espaces de représentations intermédiaires (par exemple Hough) ou encore des techniques de filtrages spécifiques. Ce sont sur ces étapes que reposent les différentes contributions du domaine que nous rappelons ci-après.

Après avoir présenté quelques techniques de référence utilisées dans le domaine de la détection temps réel de fissures, nous présentons un tableau récapitulatif contenant les algorithmes les plus importants, c'est à dire essentiellement ceux portant sur la segmentation et la classification (table 2.1).

2.2.1 Techniques bas-niveau

La plupart des techniques de l'état de l'art effectue les traitements directement sur l'image en niveaux de gris. Dans la plupart des cas, cette transformation se fait simplement à partir des canaux R(rouge), V(vert) et B(bleu) de l'image. L'image en nuance de gris représente l'intensité lumineuse des pixels. D'autres représentations existent mais elles ne seront pas abordées ici. Ce choix est justifié par la définition d'une fissure et sa manifestation, essentiellement centrée autour du contraste et de la notion d'intensité lumineuse. De plus, la conversion des images vers d'autres représentations et leurs

exploitations entraîneront des coûts plus élevés en temps de calcul. Ainsi, les modèles tels que HSV, YUV ou Lab ne présentent pas d'avantage particulier pour la détection de fissures.

Dans les années 2000, les premières techniques de détection de contours ont été développées. Dans [Ito02], les auteurs utilisent un seuillage à 2 niveaux. D'abord un seuillage fixe sur toute l'image puis un seuillage par bloc 9x9. Ensuite, les auteurs effectuent un suivi des pixels détectés pour assurer la continuité de la fissure. La recherche se focalise sur les pixels directement adjacents. Dans [Abdel-Qader03], ils présentent plusieurs techniques bas-niveau de détection de contours. Parmi elles, les méthodes classiques de détection de Sobel et Canny sont adoptées. Elles restent la référence jusqu'en 2014. Dans [Zhang14], les auteurs emploient le seuillage automatique de Otsu après avoir lissé l'image avec un filtre moyenneur. La méthode d'Otsu consiste à trouver le meilleur seuil qui maximize la séparation entre les 2 types de pixels. Ensuite des opérations morphologiques sont effectuées afin d'exploiter l'image segmentée dans une classification par Extreme Machine Learning ou par seuillage.
Les auteurs de [Pereira15] utilisent le filtre de Sobel comme algorithme de détection de contours. L'opérateur de Sobel consiste à calculer le gradient d'une image de manière discrète. L'opérateur est généralement divisé en 2 filtres de convolution (selon l'axe x et selon l'axe y). Ensuite, la magnitude est calculée en sommant des valeurs absolues ou la norme L2 des vecteurs selon les deux axes. Ce filtre est utilisé dans plusieurs articles de détection de fissures. Dans [Fujita10], les auteurs effectuent un filtre médian puis, après différence avec l'image d'origine, obtiennent une image qui est alors segmentée par la méthode d'Otsu. Ensuite, la méthode de relaxation probabiliste est exploitée pour la détection.
Toujours en exploitant le seuillage, [Phung17] utilise une technique de seuillage basée sur la moyenne et la déviation standard sur une fenêtre donnée. [Dorafshan18] compare des techniques de détection de contours avec des techniques d'apprentissage. Les auteurs affirment que les techniques bas niveau ont un rôle important dans la segmentation des fissures du fait de la nécessité de précision locale autour des informations pixellaires.

L'ensemble de ces techniques bas-niveau sont généralement très rapides en temps de calcul et ont des résultats variables selon la prise de photographies. Elles sont généralement les plus utilisées et donnent des résultats intéressants. Malheureusement, peu d'auteurs précisent la résolution des images utilisées ou la taille des fissures détectées en pixels, ce qui pour notre application est important car conditionnant deux enjeux majeurs : les conditions d'acquisition et la construction du détecteur de fissures.

2.2.2 Domaine fréquentiel

Les algorithmes dans le domaine fréquentiel ont été très exploités notamment pour leur rapidité de calcul. Les calculs sont faits dans le domaine fréquentiel après transformation des données du domaine spatial. La transformation inverse est importante pour une localisation précise.
Dans [Abdel-Qader03], les auteurs présentent la transformée en ondelettes de Haar et la transformée de Fourier. Les ondelettes de Haar permettent la décomposition d'une image en ses différentes parties basse fréquence et haute fréquence. La détection d'une fissure se fait généralement par l'atténuation des basses fréquences et en laissant passer les hautes fréquences. Cependant, un filtrage des fréquences ne suffit généralement pas et il faut pouvoir identifier les fissures parmi le bruit détecté. En effet, le bruit se concentre sur les hautes fréquences et il est donc nécessaire de pouvoir le distinguer. Il s'agit donc plus là d'une technique de segmentation voire de pré-traitement car, en l'état, elle ne conduit pas à l'identification claire et sans ambiguë des fissures sur l'image.

2.2.3 Techniques de détection de lignes

La transformée de Hough est un détecteur de lignes proposé en 1972 dans [Duda72]. Toutes les lignes passant par un point de l'image sont représentées dans l'espace de Hough par une sinusoïde. Dans cette espace, deux points sont reliés entre eux si leur

courbe dans l'espace de Hough se coupent. Ainsi on a une distribution probabiliste des lignes possibles dans une image.

Dans [Hu10], les auteurs proposent un détecteur de fissures avec un vecteur caractéristique basé sur la transformée de Hough. Une image est divisée en plusieurs cellules de taille identique. Sur ces dernières, les auteurs calculent une transformée de Hough pondérée par l'intensité du gradient de la cellule considérée. Puis, une décision est prise à partir des propriétés du vecteur obtenu en utilisant des techniques de machine learning.

Les auteurs dans [Gioi12] décrivent un détecteur de lignes droites nommé LSD. L'algorithme est conçu pour ne considérer aucun paramètre bien que des paramètres internes soient fixés. Il se décompose en plusieurs étapes :

1. L'image est redimensionnée à 80% de sa taille
2. On calcule le gradient de l'image avec un filtre 2x2 se rapprochant du filtre de Roberts. Les dérivées selon l'axe x et selon y sont calculées. L'angle du gradient, primordial pour la suite, est également calculé.
3. Les pixels sont triés en fonction de leur contraste (magnitude du gradient).
4. On construit un ensemble de pixels pour former une région support de ligne. Une tolérance d'angle est fixée à 22,5 degrés.
5. Ces régions sont évaluées et approximées en rectangle correspondant à une ligne. Plusieurs améliorations sont réalisées durant cette étape notamment une validation avec le calcul du nombre de fausses alarmes associées à un rectangle.

L'algorithme est testable en ligne. Un inconvénient important est la détection double des lignes car on retrouve deux gradients de part et d'autre d'une ligne sombre sur fond clair ou inversement.

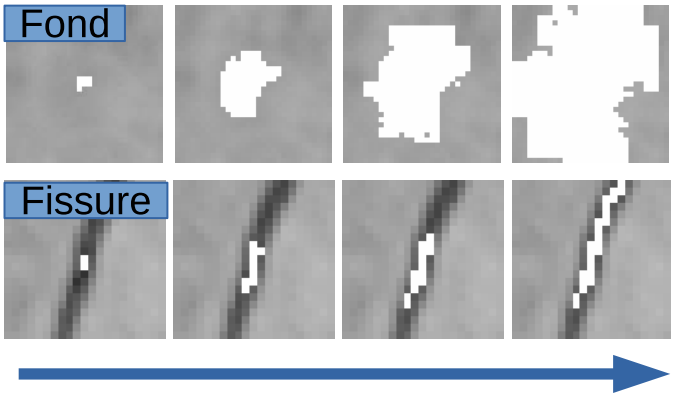


FIGURE 2.2 – Exemples du processus du percolation ([Yamaguchi09])

2.2.4 Autres techniques

Percolation

Les méthodes de percolation ont été proposées par [Yamaguchi09] dans le but de détecter presqu'en temps réel les fissures dans de grandes images. A l'origine, la percolation est un phénomène physique qui désigne le mouvement d'un liquide à travers une surface poreuse sous l'effet notamment de la pesanteur. En mathématiques, le phénomène représente alors la transition d'une information relayée, ou pas, au fur et à mesure du déplacement de l'information. Ici la fissure est l'eau par analogie et la surface est l'image de mur ou de route par exemple. On va chercher, à partir de points potentiels fissures, à propager cette dernière en analysant le niveau de gris de l'image. Des critères sont établis pour éviter des calculs trop longs : une étape *skip* est effectuée si la propagation retenue est trop circulaire, (voir figure 2.2).

Les techniques de percolation ont été reprises et améliorées par [Qu15].

Filtres à particules

Les filtres à particules sont utilisés principalement pour réaliser un suivi d'objets. Ces filtres apparaissent dans [Lins16] pour la détection de fissures. Initialement, 4000 échantillons sont choisis. Un modèle est basé sur la colorimétrie des pixels fissures. Les filtres à particules permettent d'estimer la distribution de probabilités d'un ensemble

d'échantillons pondérés. L'objectif est de prédire l'état X_k d'un système sachant les k observations connues. La prédiction évolue uniquement selon l'état précédent X_{k-1} et est mis à jour par une fonction de vraisemblance avec le modèle visé. Ici, le modèle est basé sur un histogramme local de la couleur de chaque pixel. Les filtres à particules sont comparés dans [Oliveira13] avec des techniques bas-niveau comme la binarisation avec la méthode d'Otsu.

SEGMENTATION	Seuillage	[K.N.SIVABALAN10]/ [Oliveira13]/ [Xue-jun Xu13]/ [Yu17]/ [Dorafshan18]
	Op. morphologiques	[Iyer06]/ [Jahanshahi11]/ [Sim13]/ [La14]
	Filtrage bas-niveau	[Fujita10]/ [Salman13]/ [Yeum15]/ [Talab16]
	Fréquentielle	[Chambon10]/ [Medina10]
	Percolation	[Yamaguchi09]/ [Zou12]/ [Qu15]
	Filtres à particules	[Pereira15]
	Autres	[Anwar14]/ [Prasanna14]
CLASSIFICATION	Graphes	[Zou12]/ [La14]
	SVM [1]	[Hu10]/ [Gavilán11]/ [Chanda14]/ [Prasanna14]
	K-means	[Oliveira13]/ [Cui14]
	CNN [2]	[Cha17a]/ [Yang17]/ [Kim18]/ [Dorafshan18]
	RCNN ou FCN [3]	[Cha17a]/ [Zhang17]/ [Yang18]
	Autres	[Zhang14]

TABLE 2.1 – Récapitulatif de quelques algorithmes de la littérature

Référence	Temps de calcul (s/pixel)	Dimension des images	Résolution
[Zou12]	1.04e-5	800x600	-
[Prasanna14]	6.7e-5	1920x1280	0.5mm/px
[Qu15]	1.6e-4	400x300	-
[Lins16]	1.02e-6	1712x1144	0.35mm/px
[Dorafshan18]	2.8e-7	2592x4608	0.12mm/px

TABLE 2.2 – Récapitulatif des temps de calcul pour la détection des fissures dans la littérature.

Les tableaux 2.1 et 2.2 présentent un récapitulatif des techniques employées dans la littérature concernant la détection de fissures. Peu d'articles expriment les temps d'exécution. Les résultats des temps ne sont pas issus de réimplémentations de notre part. De plus, un facteur à prendre en compte est le système sur lequel s'exécute les algorithmes. Ces valeurs ne permettent donc d'avoir qu'une idée très approximative du temps d'exécution.

2.2.5 Les approches par apprentissage

Nous constatons également l'effervescence des techniques d'apprentissage profond pour cette phase de détection de fissures, [Cha17a] [Yang18] [Zhao17] [Kim18]. Cependant, aujourd'hui, aucune approche ne peut prétendre être en mesure de faire du temps réel sur de très grandes images. Les résultats sont toutefois très intéressants du point de vue de la précision de détection et de la tolérance aux variations de situations rencontrées. Elles sont en effet capables de détecter des fissures fines ou épaisses avec le même paramétrage.

2.2.6 Détection de fissures par points d'intérêt

Après avoir présenté quelques unes des principales approches de détection de fissures de la littérature, nous allons maintenant introduire une autre approche possible de la détection de fissures, qui sera précisément celle que nous avons adoptée. En effet, plutôt que de rechercher une fissure dans son ensemble, nous proposons de rechercher des zones ponctuelles ou presque, susceptibles de recomposer une fissure lorsqu'on les associe. Ces zones ponctuelles seront assimilées à des points de l'image ayant les "bonnes" caractéristiques pour un élément constitutif d'une fissure. Nous définissons ainsi un cas particulier de ce que l'on désigne plus généralement par point d'intérêt. C'est donc sur des points d'intérêt aux caractéristiques particulières que nous baserons notre contribution à la détection de fissures.
Nous présentons donc ici certains algorithmes de détection de points d'intérêt en temps réel car ils permettront d'éclairer le fonctionnement de nos points d'intérêt, présentés

plus en détail par la suite.

SUSAN (Smallest Univalue Segment Assimilating Nucleus)

Ce détecteur [Smith97] est un précurseur de plusieurs détecteurs de contours ou de points.

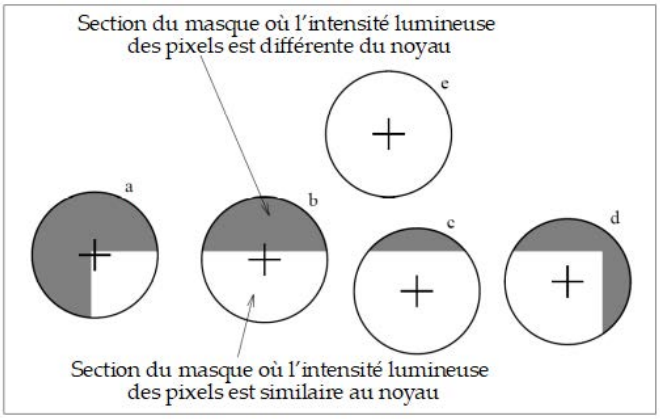
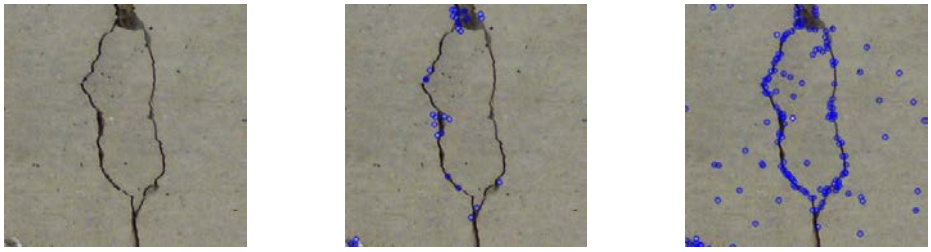


FIGURE 2.3 – Extrait de SUSAN

On applique un masque isotropique autour d'un point central (le noyau), afin de rechercher une zone similaire en intensité. Le nombre de pixels présents dans cette zone caractérise l'information structurelle de la zone de l'image. Ce nombre est nommé USAN. Plus ce nombre est élevé, plus on est dans une région homogène. Avec ce nombre, on va appliquer un seuil dépendant de la structure recherchée qui correspond à un contour ou un coin comme décrit dans [Smith97]. Pour les coins, une vérification de l'emplacement du centroïde et la contiguïté de la zone similaire est faite pour éliminer les faux-positifs. Ce détecteur pouvait être utilisé comme détecteur de contours tout comme détecteur de point. Les auteurs montrent que leur méthode donne une très bonne précision sur la localisation des points détectés.

FAST

Plus tard, une implémentation de SUSAN sur la détection de coins connaît un grand succès. FAST peut être vue comme une implémentation optimisée de SUSAN pour la détection des coins. Au lieu de comparer les pixels de toute zone locale autour d'un



(a) (b) (c)

FIGURE 2.4 – Exemples de détection avec des points d'intérêts. (a) Image originale (b) Détection avec SIFT (c) Détection avec FAST

noyau, on analyse un nombre réduit de pixels dans le masque. Une analyse intelligente permet d'améliorer la rapidité de détection. On peut savoir si un pixel ne correspond pas à un coin en comparant uniquement 4 pixels autour du pixel central. Cela est très utile pour de la détection en temps réel. D'autres améliorations portant sur les temps de calcul ont fait leur apparition, notamment par l'utilisation d'un arbre de décision construit par apprentissage sur plusieurs images. Plusieurs variantes sont disponibles et peuvent présenter des invariances à la rotation (Oriented Fast) ou encore opérer une détection à plusieurs échelles comme BRISK [Leutenegger11].

La figure 2.4 montre des exemples d'application des détecteurs de points d'intérets dans le but de détecter des fissures. On constate le potentiel des points FAST. La nature de la fissure permet une détection de pixels sur la fissure qui correspond à des coins. Notre contribution peut être vue également comme une implémentation de SUSAN concernant la structure particulière des lignes. Notre algorithme est inspiré de FAST dans le calcul optimisé. Quelques étapes supplémentaires sont proposées dans la section suivante en vue d'une bonne détection de fissures.

2.3 Pré-traitement : super-résolution

Tout au long de ce travail de détection de fissures, nous avons été confrontés au problème de la qualité et de la résolution des images utilisées. En effet, la dimension des fissures les plus fines que nous nous devions de détecter était en dessous de la

dimension d'un pixel, ou tout juste à la limite. Plusieurs pré-traitements ont donc été envisagés afin d'améliorer le processus de détection de fissures. Nous présentons ci-dessous le principe de super résolution qui peut servir à affiner la détection si on n'a pas de contraintes de temps.

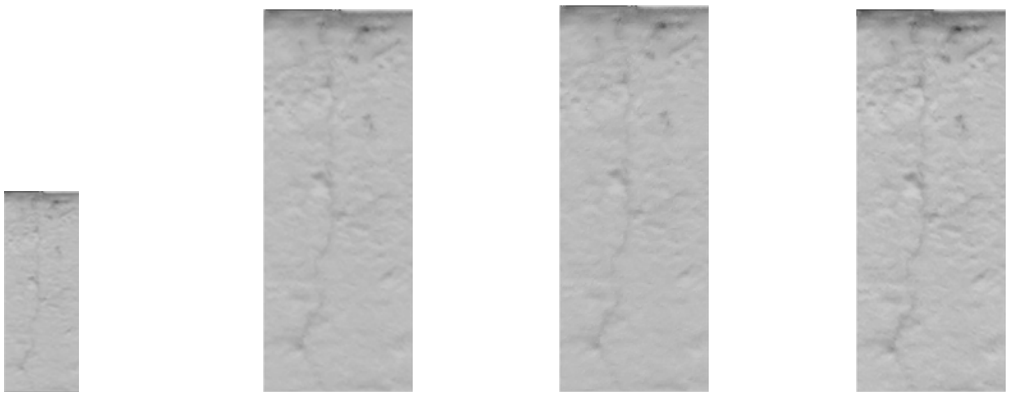
On constate souvent qu'une partie de fissure peut être plus contrastée sur une image et non visible sur une autre image. La détection d'une fissure peut donc être incomplète si l'on ne considère qu'une seule image. Une acquisition multi-images et les techniques de super-résolution permettraient de construire une meilleure image de la scène visée en construisant une image à plus grande résolution.

L'utilisation de plusieurs images de la même scène a permis à Nasrollahi dans [Nasrollahi14] de s'orienter vers une analyse multi-images et d'exploiter des techniques de Super Résolution afin d'aider la détection. Ces dernières permettent d'augmenter la résolution de l'image sans réaliser d'interpolations sur les pixels manquants. Il s'agit d'exploiter au maximum les informations à disposition pour reconstruire une image de plus grande résolution fidèle à la réalité de la scène. Dans ce domaine, les techniques fréquentielles ont été testées car elles sont plus rapides et permettent en même temps de faire de la détection de contours. On retrouve notamment des techniques basées sur la transformée en ondelettes de Haar (ou de Daubechies) [Nievergelt13] ou la transformée de Fourier (méthode de Papoulis-Gerchberg [Jain04]). On constate que l'image résultante de ces différents algorithmes s'est révélée le plus souvent soit trop bruitée, soit trop interpolée (l'image ne rend pas la vérité terrain). Des exemples d'illustrations sont présentés sur la figure 2.5.

Ce pré-traitement n'a pas été retenu dans notre application pour 2 raisons principales :

— Le *temps de traitement* est un facteur important dans le processus de détection de fissures en temps réel. Les ressources embarquées limitent l'utilisation de ces algorithmes actuellement.

— *L'interpolation de l'image à haute résolution* est un problème car le bruit (ou imperfection) du mur peut entraîner à plus grande résolution la génération de fissures



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

FIGURE 2.5 – Exemples de techniques de super-résolution. (a) Image basse résolution (b) Image haute résolution (HR) interpolation bicubique (c) Image HR ondelettes de Haar (d) Image HR méthode Papoulis-Gerchberg

non existantes.

2.4 Notre contribution : Les points FLASH

A l'origine, l'étude que nous avons menée était soumise à un certain nombre de contraintes, et parmi celles-ci, il y avait le temps de traitement qui devait être "temps réel" ou proche temps réel, car cette partie d'analyse d'images s'intègre à un processus plus complet de contrôle de qualité par drones. Notre algorithme est donc d'abord conçu pour répondre à cette contrainte forte de détection en proche temps réel, et le choix des approches algorithmiques vont dans ce sens. Par ailleurs, il faut noter que nos images cibles sont des images de béton lisse d'un gris uniforme et très peu constrastées. Il s'agit là d'une spécificité forte de nos images.

2.4.1 Principe : détection de points fissure

L'objectif de l'analyse automatique des images d'ouvrages d'art est la détection automatique des défauts de la structure. Parmi l'ensemble des défauts possibles, le premier auquel nous nous sommes intéressés, est la fissure. Notre approche procède

par accumulation d'évidences. Nous commençons par détecter dans l'image des zones, ou pixels, qui présentent les caractéristiques voulues pour être un élément constituant d'une fissure. Puis nous analysons les liens qui existent entre ces différentes zones candidates. Si leurs relations mutuelles s'organisent de façon cohérente avec les critères attendus pour former une fissure, nous les agglomérons les unes aux autres, et cela de façon itérative (voir la section 2.4.6).

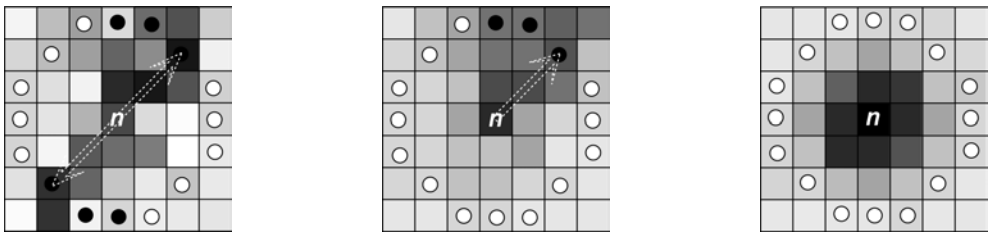
Notre détecteur est basé sur la caractérisation de points dans une zone réduite autour d'un point clé. Pour des raisons de symétrie, puisqu'il n'y a pas de directions à privilégier dans cette analyse, la zone en question sera circulaire. Nous avons testé pour cela plusieurs ensembles de points possibles. Les cercles de Bresenham de rayon 3 et 2 ont retenu notre attention. Ce sont des cercles qui sont définis sur un espace discret. Pour le choix des rayons, nous avons opté pour un compromis entre importance de la zone analysée et temps de traitement. Plus le rayon est grand, plus l'analyse devient consommatrice pour une détection temps réel. Au cours de cette analyse, nous distinguons trois types de configurations locales, ou caractéristiques locales des pixels de l'image, ce qui nous a conduit à définir trois types de points, listés ici par ordre décroissant d'importance dans la constitution des fissures :

— les points de type *micro-lignes*
— les points de type *coins*
— les points de type *bulle*

Ce sont ces différents types de points, illustrés dans la figure 2.6, que nous désignons sous le nom de points FLASH pour *Fast Local Analysis by threSHoldig*. La détection des points de type micro-lignes est donc la première phase essentielle dans notre détection des fissures.

2.4.2 Points de type *micro-lignes*

On considère une image dans sa version en niveaux de gris. Tous les pixels de l'image sont testés les uns après les autres. On peut d'ailleurs envisager de n'en tester qu'un sur deux en suivant une répartition en forme de damier par exemple, pour accé-



(a) (b) (c)

FIGURE 2.6 – Exemples des différents types de points détectés. (a) point "micro-ligne" (b) point de type "coin" (c) point "bulle"

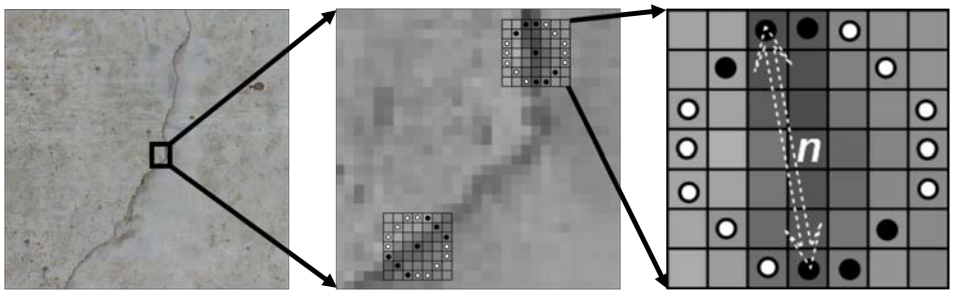


FIGURE 2.7 – Masque du détecteur. Pour que le pixel n soit retenu, il faut que les pixels identifiés par les points blancs soient plus clairs et que les pixels identifiés par les points noirs soient plus sombres que les pixels clairs. Ici, ces pixels se trouvent sur un cercle de Bresenham de rayon 3.

lérer l'analyse. Lorsqu'on arrive sur le point n (que l'on retrouve sur l'image de droite de la figure 2.7), on récupère $I(n)$, son intensité de niveau de gris. Plusieurs tests vont alors être réalisés, en plusieurs étapes, entre cette valeur $I(n)$ et les différentes valeurs de niveau de gris $I(p)$ de différents pixels p le long du périmètre d'un cercle de Bresenham. Pour retenir le point n comme un point intéressant et pouvant potentiellement appartenir à une fissure, il faudra que :

$$\forall p \in P_{white}, I(p) >= I(n) + t \tag{2.1}$$

avec ici

— P_{white} qui est un ensemble de pixels formé par 5 pixels consécutifs sur le périmètre du cercle de Bresenham ainsi que les 5 pixels qui sont leurs symétriques.

Ces pixels sont identifiés par des cercles blancs sur la figure 2.7. Sur cette figure, on ne représente qu'une des configurations possibles. Toutes les configurations que l'on peut obtenir par rotation de cette configuration le long du cercle de Bresenham seront aussi testées.

— t qui représente une valeur d'offset qui permet de régler l'importance du contraste que l'on impose à un pixel par rapport à son voisinage pour être un pixel de fissure.

Si le point n ne remplit ces conditions pour aucune des différentes configurations possibles, l'analyse s'arrête là pour ce point et on passe au pixel suivant. Il faut noter qu'il n'est pas nécessaire de tester de façon exhaustive, l'ensemble des configurations P_{white} possibles pour pouvoir décider ou non d'arrêter l'analyse. En effet, lorsqu'une configuration n'est pas possible, on peut en éliminer 5 autres. Si en revanche une configuration P_{white} vérifie cette condition, cela signifie que le point n a un niveau de gris $I(n)$ significativement inférieur à l'ensemble des voisins P_{white}. L'importance attribuée au terme *significativement* précédent est justement donné par la valeur de l'offset t précédente. Ce dernier peut être fixé par toutes les valeurs de niveaux de gris possibles. Le faible contraste des fissures fines nous a poussé à fixer t à 5. Une plus grande valeur permet d'éliminer le bruit mais réduit la capacité de détection des fissures peu contrastées.

L'analyse continue alors et on passe aux tests sur les points P_{black} et P^s_{black}, pour leur symétriques. Cette deuxième série de tests vérifie si :

$$\exists p_b \in P_{black}, I(p_b) < \min(I(p_w)) \text{ avec } p_w \in P_{white} \tag{2.2}$$

et

$$\exists p^s_b \in P^s_{black}, I(p^s_b) < \min(I(p_w)) \text{ avec } p_w \in P_{white} \tag{2.3}$$

Si ces conditions sont vérifiées, le point n est alors considéré comme un point de la catégorie des *micro-lignes*. Il s'agit d'un point qui se trouve au centre d'un cercle de Bresenham de rayon 2 ou 3 (la figure 2.7 illustre la situation d'un cercle de rayon 3 utilisé dans l'essentiel de nos tests) pour lequel les deux zones opposées par symétrie centrale

sont significativement plus sombres que le reste du périmètre. La valeur de l'intensité $I(n)$ associée à l'offset t sert de seuil de référence. De tels pixels n sont potentiellement les indicateurs d'une fissure à cet endroit.

2.4.3 Points de type *coins*

Cette détection de fissure doit s'accompagner de la détection de ses extrémités. Les micro-lignes précédentes ne correspondent pas à ces extrémités. En effet ils indiquent le passage d'une fissure. Une extrémité se manifestera plus directement par ce que nous appellerons un point de type *coin*, au sens d'un point FAST. A la différence des points de type micro-lignes précédents, ces points coins correspondent à la situation où il n'y a qu'une seule des deux zones opposées par symétrie centrale qui est significativement plus sombre que le reste du périmètre. En reprenant nos notations, nous pouvons l'exprimer par la condition suivante (les deux ensembles P_{black} et P^s_{black} jouent ici un rôle symétrique, nous supposons que c'est P^s_{black} qui contient au moins un point plus sombre) :

$$\begin{aligned} &\forall p_b \in P_{black}, I(p_b) >= \min(I(p_w)) \\ &\text{et } \exists p^s_b \in P^s_{black}, I(p^s_b) < \min(I(p_w)) \text{ avec } p_w \in P_{white} \end{aligned} \tag{2.4}$$

2.4.4 Points de type *bulles*

Enfin, la dernière situation possible, qui complète les deux précédentes, définira les points de type *bulles*. Ces points correspondent aux situations où aucune des deux zones opposées par symétrie centrale n'est significativement plus sombre que le reste du périmètre. Le point central est alors entouré de pixels significativement (au sens de notre seuil t) plus clairs. Ces points de type *bulles* n'ont pas d'intérêt dans cette partie de détection et de caractérisation de fissure. Cependant, ils trouveront toute leur utilité un peu plus loin, lors de la phase d'appariement d'images que nous réaliserons à partir de ces calculs de points FLASH, déjà extraits ici pour la détection de fissures. En reprenant

nos notations, ces points de type *bulles* se caractériseront par :

$$\forall p_b \in P_{black} \cup P^s_{black}, I(p_b) >= \min(I(p_w)) \text{ avec } p_w \in P_{white} \tag{2.5}$$

Il s'agit donc de points sans aucune orientation ou direction spécifique calculable mais cela ne nous empêchera pas de les utiliser en complément dans le processus d'appariement d'images.

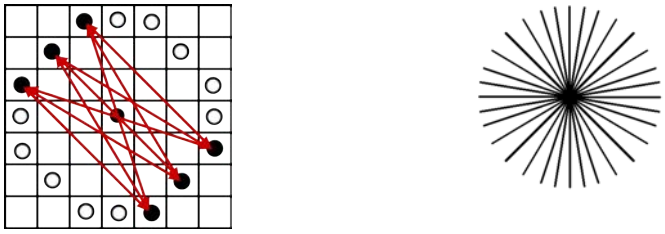
Tout comme FAST, le nombre réduit de pixels à analyser et la configuration recherchée permettent d'améliorer les temps de calcul. Pour éliminer un pixel rapidement, on peut analyser uniquement 2 pixels et leur symétrique : c'est à dire 2 segments. Selon les tests de segments, le nombre de configurations possibles pour les micro-lignes est mis à jour. Tous les tests ne sont donc pas réalisés. Si aucune configuration n'est possible, on évalue le pixel suivant.

2.4.5 Détermination d'un score de contraste et de l'angle caractéristique

Á ce stade il est nécessaire d'associer à chaque point fissure détecté un score S de contraste, qui permet d'avoir une évaluation du contraste de la fissure potentiellement associée, relatif à sa visibilité. On le définit de la façon suivante :

$$S = \min_{p \in P_{white}} I(p) - I(n) \tag{2.6}$$

Puis, l'orientation est calculée. Il correspond à l'orientation du vecteur qui relie le point le plus sombre de l'ensemble P_{black} au point le plus sombre de l'ensemble P^s_{black} de leurs symétriques (dans le cas de la recherche d'une fissure sombre sur fond clair). En considérant le cercle de Bresenham de rayon 3, l'orientation calculée n'est pas uniforme. On a 16 orientations possibles avec un écart maximal de 14 degrés et un écart minimal de 9 degrés (voir figure 2.8).



(a) (b)

FIGURE 2.8 – Calcul de l'orientation d'un point "micro-ligne". (a) Toutes les orientations possibles dans la configuration détectée. (b) Toutes les orientations possibles en considérant toutes les configurations.

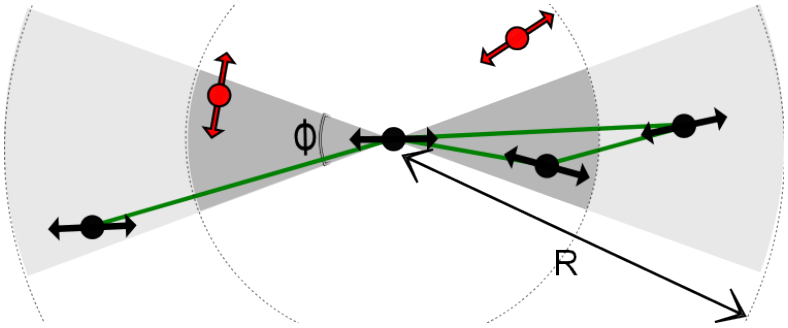


FIGURE 2.9 – Détection de lignes et constitution de graphes

2.4.6 Constitution des fissures

Le suivi est inspiré de l'algorithme LSD ([Gioi12]). Nous avons introduit plusieurs niveaux de suivi d'une fissure. En effet, les points fissures ne sont pas nécessairement adjacents. Il faut donc effectuer une analyse à plus grande échelle et considérer un ensemble de points. Nous définissons une région, plus précisément un disque de rayon R autour d'un point central. Cette région est divisée en n intervalles réguliers. Sur la figure 2.9, on peut distinguer deux intervalles. Dans ces derniers, on va chercher à construire un ensemble de points joignables, c'est-à-dire un ensemble de points qui, bien que non adjacents, peuvent néanmoins être considérés avec une forte probabilité d'appartenance à une même fissure. Plusieurs stratégies peuvent être adoptées. Ici nous décrivons deux stratégies correspondants à deux applications.

— Recherche de lignes droites

La recherche de lignes droites permet de restreindre l'angle de recherche Φ de points distants à joindre (ou joignables). Pour un point retenu comme point potentiellement fissure, dont on connaît donc l'orientation, on ne cherchera à le relier qu'à d'autres points fissures situés dans l'axe de son orientation. Nous pouvons alors aller plus loin en distance dans la recherche de points joignables sans perdre en temps de calcul. Par exemple, les lignes droites peuvent servir à détecter des jonctions entre murs. La zone de recherche de points joignables est subdivisée en 4 intervalles de plus en plus éloignés. Pour rechercher des points joignables dans l'intervalle d'éloignement supérieur suivant, un nombre minimal de points est requis. Nous écartons ainsi plusieurs éléments parasites ainsi que les points détectés isolés.

— Recherche de fissures

Le caractère aléatoire de la direction d'une fissure doit être pris en compte. En effet, il n'y a que la connaissance des forces s'exerçant sur l'ouvrage qui permette de connaître le sens de propagation globale d'une fissure dans le matériau. Mais localement, la fissure peut prendre à peu de choses près n'importe quelle direction. C'est pourquoi, plus de tolérance est nécessaire pour les fissures et nous

ne pouvons pas aller très loin du point courant car les orientations peuvent vite changer. Ici trois intervalles sont utilisés et le nombre de points requis est le même que précédemment. L'angle de tolérance Φ peut varier entre 25 et 45 degrés.

Nous privilégions la recherche de fissures mais la recherche de lignes droites peut servir à détecter par exemple des jonctions entre deux murs. Une fois les points joignables définis pour chaque point détecté, on peut construire des graphes de la façon suivante : Notons p un point et τ_p son ensemble de points joignables. Une liaison est validée entre un point a et un point b si :

$$a \in \tau_b \quad \text{et} \quad b \in \tau_a \tag{2.7}$$

Plus formellement, un point détecté v_i forme d'abord un graphe simple orienté noté $g_i = (V_i, E_i)$ avec $E_i = \{(v_i, v_j)\}_{i \neq j, v_j \in V_i}$. L'ensemble des sommets V_i est donc l'ensemble des points joignables $\{v_j\}$ et le point considéré v_i. Puis on cherche à créer un graphe simple non orienté et on conserve les arêtes si pour deux sommets v_i et v_j rattachés à leur ensemble d'arcs E_i et E_j, on a :

$$(v_i, v_j) \in E_i \quad \text{et} \quad (v_j, v_i) \in E_j \tag{2.8}$$

Après cette opération, nous obtenons des graphes non orientés. Les sommets x ayant pour degré $d(x) = 0$ sont éliminés.

Les points n'ayant aucune connexion sont alors supprimés améliorant ainsi la résistance au bruit. A la fin du processus, un ensemble de graphes qui modélise les relations entre chaque pixel fissure est obtenu et constitue l'ensemble de fissures ou morceaux de fissures potentielles. A l'aide de ces graphes, nous pouvons alors effectuer d'autres améliorations. Ces graphes sont transformés en arbre couvrant par un parcours en largeur d'abord du graphe afin de connaître les points/sommets constituant les graphes.

2.4.7 Optimisation de la recherche des points

Pour effectuer une recherche rapide des points voisins autour d'un point central, une structure de données particulière a été utilisée. Une structure matricielle de la taille de l'image a été retenue et nous sauvegardons uniquement les points FLASH. Dans les données enregistrées, la position du prochain point fissure détecté est présente en chaque point. S'il y a présence de point détecté alors les informations concernant son orientation, son score, ses points joignables sont ajoutés dans la structure. La recherche de points est réalisée par un balayage en spiral dans une fenêtre prédéfinie. La complexité de cette recherche est alors en $O(n)$ pour n le nombre de points détectés.

2.4.8 Représentation et calcul de la longueur d'une fissure

Chaque graphe ainsi constitué est une fissure potentielle. Une fois ces graphes de points constitués, nous pouvons les analyser afin de réduire les faux positifs, c'est à dire éliminer les regroupements de points qui ne sont pas des parties de fissures véritables. L'analyse réalisée ici conduit également à la construction d'une représentation simplifiée de la fissure.
Nous commençons par détecter la direction principale de la fissure à partir de la direction principale du graphe qui peut être obtenue de deux manières différentes. Selon la contrainte de temps dont on dispose, on pourra choisir l'une des deux méthodes :

Méthode 1

Le calcul se fait simplement en prenant la moyenne des angles caractéristiques de tous les points présents dans un graphe (les sommets du graphe). Cependant, la moyenne ne peut pas se calculer comme une simple moyenne arithmétique ; il faut prendre en compte le fait que l'on considère des directions et non des orientations. La moyenne d'angles entre un point ayant un angle de 0 degré et un point ayant un angle de 160 degrés n'est pas 80 mais 170 degrés. On utilise le calcul de direction linéaire moyenne (LDM) décrit dans [Mitchell20] qui utilise le cosinus et le sinus (par coordonnées polaires) pour avoir une moyenne d'angles de tous les points.

Méthode 2

On utilise les techniques d'Analyse en Composantes Principales (ACP). Avec un nuage de points simples dans un espace à 2 dimensions, le calcul est assez rapide. Il faut d'abord calculer la matrice de covariance comme suit :

$$cov(G_i) = \begin{pmatrix} \sum x^2 & \sum x \cdot y \\ \sum x \cdot y & \sum y^2 \end{pmatrix} \tag{2.9}$$

avec G_i le graphe numéro i détecté dans l'image et constitué des points p de coordonnées (x, y). Les sommes sont effectuées sur l'ensemble des points p du graphe G_i. La recherche des valeurs propres et des vecteurs propres est triviale dans ce cas de figure. On a une matrice symétrique à 2 dimensions. Les valeurs propres sont définies par :

$$\lambda_{1,2} = 1/2 \cdot [(\sum x^2 + \sum y^2) \pm \sqrt{(\sum x^2 - \sum y^2)^2 + 4 \cdot \sum x \cdot y^2}] \tag{2.10}$$

L'orientation principale est donnée par le vecteur propre $\vec{R}$ associé à la plus grande valeur propre que l'on peut calculer de la façon suivante :

$$\vec{R} \begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{\sum x^2 \cdot \sum y^2 - \sum x \cdot y^2} \begin{pmatrix} -(\sum x^2 - \sum x \cdot y) \cdot \lambda \\ (\sum x^2 - \sum x \cdot y) \cdot \lambda \end{pmatrix} \tag{2.11}$$

Avec cette orientation principale, on va projeter les points sur la droite dirigée par le premier vecteur propre et passant par le barycentre de l'ensemble des points p du graphe. On cherche les points les plus éloignés se projetant sur cette droite et on subdivise le segment obtenu par le nombre n que l'on souhaite. Ce nombre n correspond au nombre de points conservés pour représenter la fissure. On peut ainsi garder un point représentatif par section et avoir une représentation simplifiée mais fidèle au tracé de la fissure détectée. Cette représentation pourra servir à stocker les fissures sans conserver une image complète. La représentation simplifiée de la fissure permet également d'estimer les extrémités d'une fissure (voir figure 2.10).

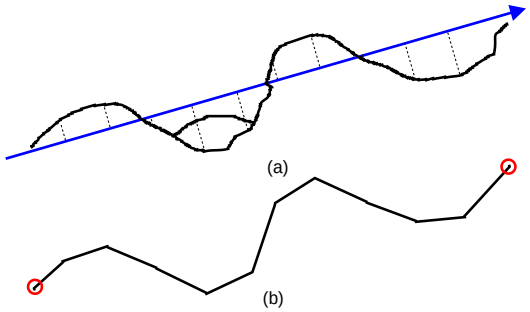


FIGURE 2.10 – Représentation simplifiée d'une fissure détectée. (a) Projection sur la droite suivant l'orientation principale de la fissure. (b) Représentation simplifiée de la fissure. En rouge les extrémités trouvées.

2.4.9 Reconstruction d'une fissure à plus grande échelle : Fusion de graphes

L'algorithme présenté ici ne rentre pas dans le cadre de l'application en temps réel et ne fait pas l'objet d'une évaluation quantitative. Il peut être utilisé si les contraintes de temps ne sont plus imposées.

Nous constatons que plusieurs petits graphes représentent une même fissure (voir la figure 2.11 - b). La détection n'est pas optimale car le long d'une fissure, les pixels fissures ayant un contraste très bas empêchent une bonne détection. Il faut alors pouvoir faire la jonction avec les autres graphes détectés.

```
Data : G : ensemble des graphes détectés par FLASH
g ← meilleur_potentiel_fissure(G)
while g ≠ dernier_element_de(G) do
    for p ∈ extremites(g) do
        while limites non atteintes et NON jonction do
            e ← chercher_meilleur_extremite(g)
            jonction ← joindre(g, graphe_de(e))
            if jonction then
                MAJ(G, e)
    if NON jonction then
        g ← meilleur_apres(G, g)
    jonction ← false
```

La fonction *meilleur_potentiel_fissure* permet de trouver le graphe ayant la plus forte probabilité d'être une fissure. Pour plus de simplicité, nous pouvons par exemple choisir le graphe le plus long, c'est à dire celui qui traduit la fissure de plus grande lareur sur l'image.

Les extrémités sont obtenues grâce aux calculs présentés dans la section précédente.

La fonction *chercher_meilleur_extremite* permet de chercher dans un cône défini par l'orientation et la longueur du graphe une autre extrémité d'un graphe différent potentiel pour faire une fusion.

La fonction *extremites* retourne les deux extrémités d'un graphe donné.

MAJ permet d'éliminier le graphe associé au point e. Ainsi, on effectue une jonction sur le même meilleur graphe tant qu'il existe à proximité des graphes joignables. Si ce n'est pas possible, on cherche le meilleur graphe après celui qu'on vient de traiter. Cette opération est réalisée simplement en utilisant une liste triée.

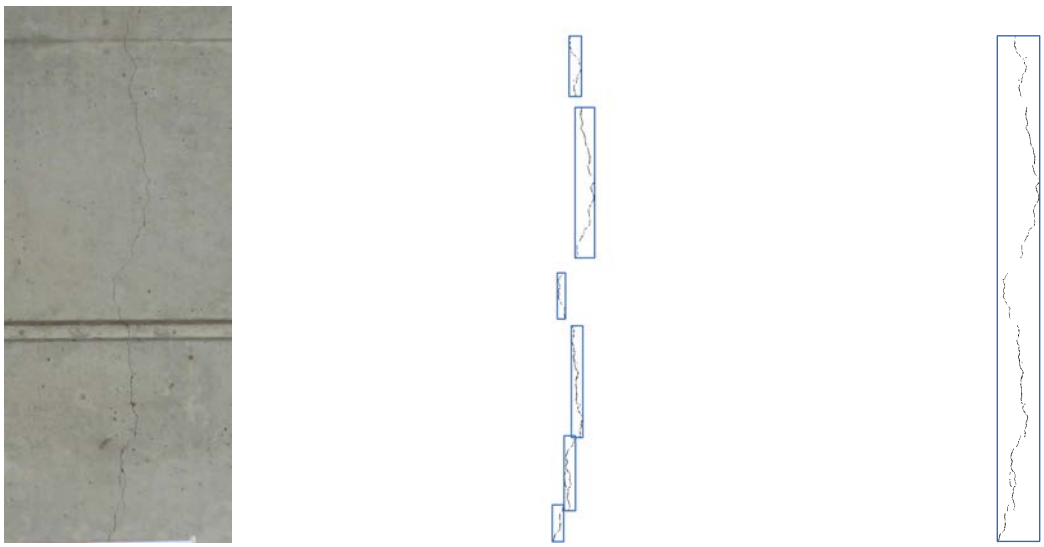
A la fin du processus, on obtient une fusion de graphes appartenant potentiellement à la même fissure. Un exemple de fusion est présenté sur la figure 2.11.

2.4.10 Constitution d'une base de donnée : FineCracks

Dans la littérature, très peu de bases de données sont disponibles. Beaucoup de travaux sont issus de collaboration avec des entreprises et par conséquent, les bases de données ne sont pas rendues publiques. Les images disponibles ne correspondent pas exactement à notre problématique. En effet, nous cherchons à évaluer la capacité à détecter en temps réel des fissures fines et non des structures très visibles.

C'est pourquoi nous avons construit une petite base d'images [4]. Cette dernière est composée d'images avec fissures provenant de mur en béton plus ou moins lisse, mais aussi d'images sans fissure. Des images avec des graffitis sont également présentes avec ou sans fissure. Puisque nous voulons utiliser cette base pour réaliser nos tests de détection et de reconstruction de fissures, nous avons ajouté à cette base une vérité terrain

4. FineCracks est disponible en ligne https ://perso.liris.cnrs.fr/yannick.faula/



(a) (b) (c)

FIGURE 2.11 – Exemples de techniques de super-résolution. (a) Image d'origine (b) Résultat sans fusion de graphe (c) Résultat avec fusion des graphes

pour chaque image. Cette vérité terrain a été réalisée avec l'aide d'experts du domaine de l'inspection des ouvrages d'art. La description des fissures n'est pas réalisée au pixel près. Il est difficile d'avoir une description fiable quand la fissure n'est pas très contrastée. De plus, le but n'est pas de segmenter une image de manière précise mais de déterminer la présence d'une fissure fine dans une zone de l'image. La description établie est donc basée sur les régions. La fissure est décrite avec des polylignes. Chaque point de polylignes est situé à moins d'une dizaine de pixels des vrais pixels fissures. Dans l'implémentation, nous avons choisi d'utiliser le format existant SVG pour la description de la fissure. Il permet d'avoir une description qui prend très peu de place. Le nombre de points reliant la polyligne peut être variable.

2.4.11 Evaluation

Dans les sections suivantes, nous allons présenter quelques évaluations des qualités et limitations de notre méthode FLASH par rapport à l'existant.

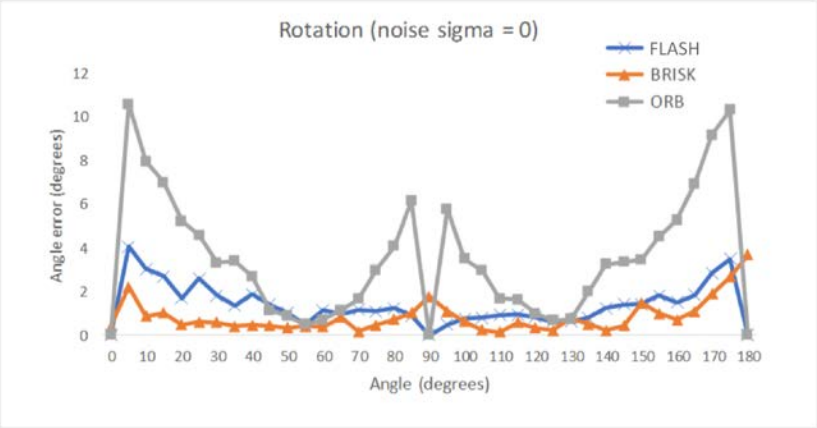


FIGURE 2.12 – Évaluation de la rotation

2.4.11.1 Robustesse de l'angle

Nous avons commencé par simuler une rotation d'une image avec un ajout de bruit gaussien. Les rotations se font par palier de 5 degrés. L'erreur mesurée est la différence d'angles entre la rotation simulée et l'angle trouvée par la détection du point d'intérêt. Les angles n'étant pas couverts complètement, une erreur nulle est impossible à atteindre. On a calculé une erreur minimale en choisissant l'angle minimisant l'erreur. Sur la figure 2.12, on peut voir que FLASH a une faible erreur d'angle similaire à BRISK. ORB est très médiocre dans la déterminatioin des angles lorsqu'il s'agit de faibles rotations, ce qui en pratique, peut régulièrement arriver sur un vecteur d'acquisition comme le drone. L'angle calculé par le détecteur FLASH est plus robuste à l'ajout de bruit gaussien que les autres méthodes. L'utilisation d'un seuil dans l'expression 2.1 permet d'augmenter cette robustesse au bruit avec une valeur t plus élevée.

2.4.11.2 Evaluation par blocs

Protocole

Dans la littérature, les évaluations sont souvent réalisées au niveau pixel. Ce niveau d'échelle n'est pas le plus adapté. En effet, nous cherchons ici à évaluer la qualité de la détection d'une fissure et non pas la qualité de la détection de tous les points d'une

fissure (autrement dit la qualité de segmentation au pixel près). C'est pourquoi nous proposons ici de réaliser une évaluation à une autre échelle que celle du pixel. Cette nouvelle échelle est plus compatible avec la représentation des fissures dans notre base de données qui est liée à une description par bloc. Ainsi l'image est subdivisée en blocs de taille MxM. Cette taille dépend notamment de la précision estimée sur l'élaboration de la vérité terrain. Si une polyligne traverse un bloc, ce bloc est considéré comme un bloc fissure. Après la phase de détection, un bloc est considéré comme présentant une fissure si on note la présence d'au moins un pixel fissure. La résultante de ces opérations est 2 matrices booléennes de taille $\lfloor H/M \times L/M \rfloor$ pour une image de taille $L \times H$. La comparaison est alors triviale et nous pouvons en déduire les nombres de

— blocs détectés fissure et labellisés *fissure*, les vrais positifs noté TP
— blocs détectés fissure et labellisés *sans fissure*, les faux positifs noté FP
— blocs détectés sans fissure et labellisés *sans fissure*, les vrais négatifs noté TN
— blocs détectés sans fissure et labellisés *fissure*, les faux négatifs noté FN

On peut alors calculer les valeurs classiques de rappel et la spécificité. Cette spécificité représente la probabilité de détecter un bloc sans fissure lorsqu'un bloc est effectivement sans fissure. On a :

$$rec_b = \frac{TP}{(TP + FN)} \tag{2.12}$$

$$spec_b = \frac{TN}{(TN + FP)} \tag{2.13}$$

En pratique, une fissure peut ne pas être détectée dans sa totalité. Les zones détectées peuvent être éparses et peuvent couvrir en longueur une bonne partie d'une fissure. Une nouvelle échelle d'analyse est requise. C'est pourquoi, une nouvelle métrique, rec_c^P, est introduite. Elle est définie comme suit :

$$rec_c^P = \frac{\text{nombre de fissures détectés pertinentes à P\%}}{\text{nombre total de fissures pertinentes}} \tag{2.14}$$

avec P le pourcentage de blocs de la fissure détectée. Ainsi, rec_b désigne le rappel à l'échelle des blocs (b pour blocs) et rec_c désigne le rappel à l'échelle des fissures (c pour *crack* c'est-à-dire fissure en anglais). Dans notre évaluation la taille M est fixée à 16 et P

est fixé à 50.

Discussion

Nous rappelons ici qu'un bon détecteur aura un rappel (rec) **et** une spécificité ($spec$) très élevés.

Des exemples de détection de fissures sur notre base de données sont présentés sur la figure 2.13. D'après le tableau 2.3, FLASH n'a pas le meilleur rappel rec_b mais est très bon sur le rec_c. Cela signifie que FLASH est capable de détecter la plupart des fissures fines et pas seulement toutes les fissures avec une grosse ouverture. Dans [Jahanshahi11], les auteurs détectent uniquement les fissures avec une grosse ouverture.

LSD n'est pas conçu pour la détection de fissures, mais l'algorithme parvient à éliminer les bruits et artefacts dans une image. Le score de spécificité le montre. En pratique, la méthode de [Pereira15] n'est pas adaptée. Son faible score de spécificité, bien que le rappel soit élevé, montre qu'il y a énormément de faux-positifs. La difficulté est de pouvoir détecter tous les défauts sans avoir une alerte sur toutes les images qui n'en présentent aucun. Cela nécessiterait une intervention humaine presque aussi importante que l'analyse de photo à la main. L'analyse spatiale des points FLASH permet d'avoir une très bonne robustesse au bruit.

Le défaut de FLASH est qu'il n'est pas multi échelle. Pour détecter les fissures de très grosses ouvertures, il faudra redimensionner l'image et refaire une passe de détection. Mais ce type de fissures assez visibles n'est pas le plus difficile à détecter sur une image. Nous nous intéressons principalement aux fissures de faibles ouvertures. En effet, rappelons que dans notre application spécifique, l'objectif est de détecter les fissures fines, de faibles ou très faibles ouvertures afin de pouvoir déclencher une nouvelle acquisition avec une résolution plus importante. Une deuxième passe "off line", c'est-à-dire une fois l'ensemble des acquisitions réalisées, peut être effectuée ensuite. Lors de cette deuxième passe, le changement de résolution pourra se faire sans être pénalisant en terme de vitesse de calcul puisque la contrainte de temps d'exécution ne sera plus présente.

Comparativement aux autres méthodes, nous pouvons fournir également des informa-

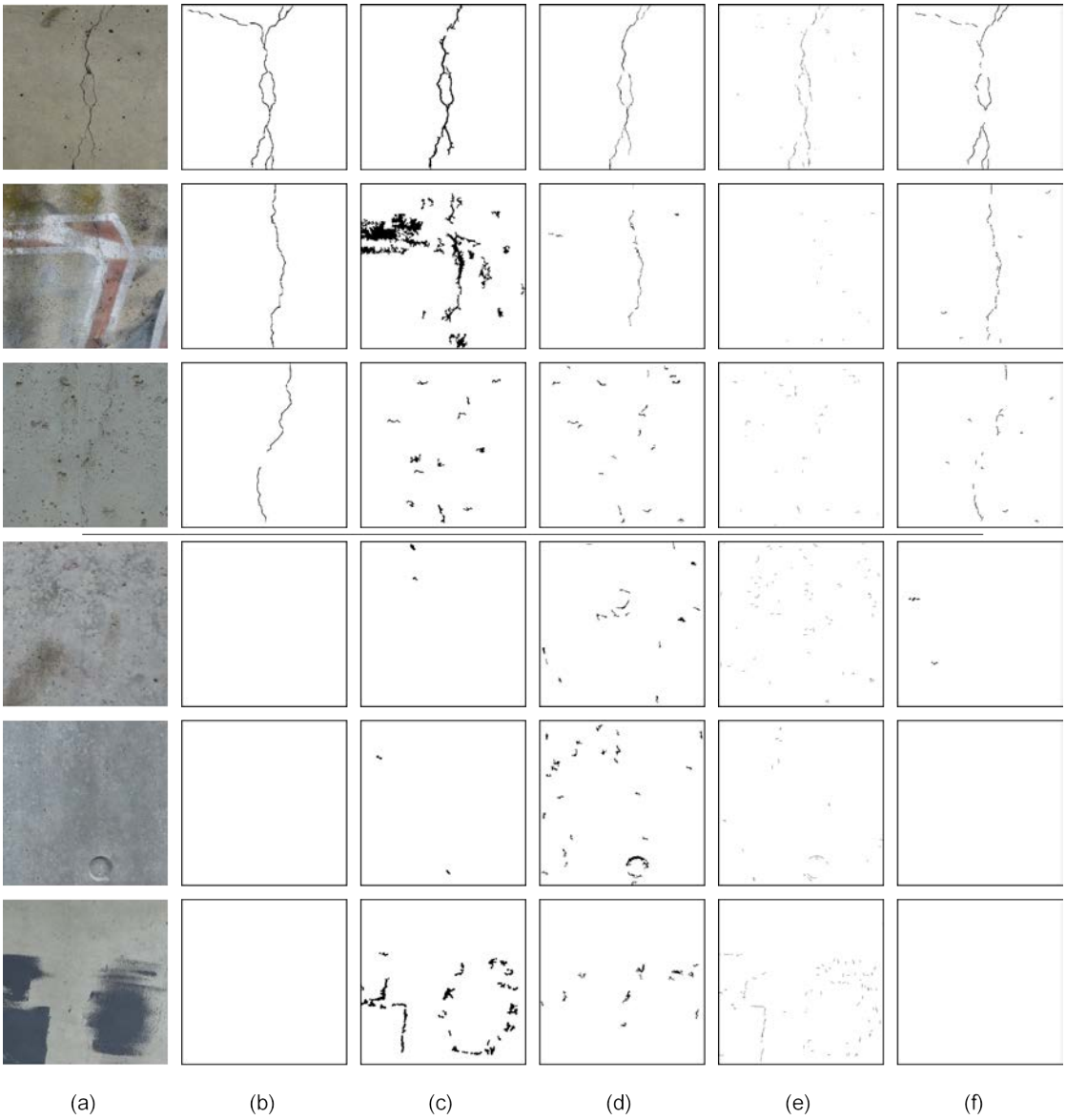


FIGURE 2.13 – Exemples de résultats de détection. (a) Image source (b) Vérité terrain (c) [Pereira15] (d) [Jahanshahi11] (e) [Gioi12] (f) FLASH

Algorithme	rec_b	$spec_b$	rec_c^{50}
FLASH	0.65	0.94	0.77
[Pereira15]	0.83	0.25	0.87
[Jahanshahi11]*	0.69	0.48	0.75
[Fujita10]*	0.74	0.41	0.80
[Gioi12]	0.25	0.98	0.12

TABLE 2.3 – Evaluation de la détection de fissures. * Implementation partielle.

Algorithme	Détection des points (10e-8 s)	Total (10e-8 s)
FLASH (t=5)	2.45	4.04
FLASH (t=10)	1.45	3.6
[Pereira15]	-	2.01
[Jahanshahi11]*	3.21	11.3
[Fujita10]*	3.32	12.1
[Gioi12]	-	11.5

TABLE 2.4 – Temps d'exécution par pixel pour la détection de fissures. * Implémentation partielle. Le traitement complet prend plus de temps.

tions relatives aux fissures. On peut envisager de trier les fissures par taille calculée ou par orientation principale.

2.4.11.3 Temps de calcul

Cette comparaison de notre méthode avec l'existant au niveau du temps d'exécution n'est pas vraiment possible. Effectivement, les implémentations des différents algorithmes de la littérature ne sont pas disponibles. De plus, certains algorithmes sont en 2 parties : une partie segmentation bas-niveau puis une partie apprentissage par machine learning sur des bases de données non disponibles, dans le but d'améliorer les résultats obtenus. Nous n'avons cherché à évaluer que la première partie (détection des points potentiellement "fissures") car nous n'avons à ce stade aucune détection par machine learning. Après la détection des points, le calcul final des algorithmes tirés de [Jahanshahi11] et [Fujita10] consiste ici à détecter les composantes connexes et à supprimer les plus petits éléments trouvés. Notre implémentation ne bénéficie pas de certaines optimisations d'OpenCV. En effet, notre structure de données est difficilement

compatible avec un calcul parallèle. Quelques parties peuvent encore être améliorées mais nécessitent plus d'investigations et du temps. Une version plus simple permet d'améliorer et/ou de modifier l'algorithme de base de notre méthode. Les algorithmes de la littérature sont un enchaînement de fonctions simples d'OpenCV. Pour toutes ces raisons, il est très difficile d'envisager une évaluation crédible de notre méthode par rapport à l'existant. Cependant, les temps présentées sur le tableau 2.4 montre bien que nous pouvons parvenir à une analyse de grandes images en temps réel.

2.4.12 Conclusion

Nous avons développé une nouvelle technique de détection de fissure quasi temps réel. Plusieurs astuces permettent d'atteindre ces temps de calcul, comme la détection de faux-positifs réduisant le nombre de pixels à analyser, mais aussi l'utilisation de structure de données intelligentes qui permet d'avoir une recherche spatiale en $O(n)$ des points détectés. Notre méthode a été inspirée par des algorithmes comme FAST et certains articles de détection de fissures. Un atout principal est de proposer une base de données pour pouvoir évaluer les algorithmes. Dans la littérature, le protocole utilisé n'est pas adapté à la pratique et ne correspond pas à l'objectif de détection de fissures. Toute cette partie de détection de fissure ne peut s'envisager que si l'acquisition photo est de qualité. Si la qualité baisse, la détection sera plus difficile. C'est pourquoi le chapitre suivant fait le point sur la qualité de l'acquisition, indissociable d'une bonne détection.

3

Détermination de la qualité de l'acquisition

3.1 Introduction

La détection de fissures fines est possible par une acquisition d'images de qualité suffisante. Une image de mauvaise qualité est difficile à traiter et les détériorations n'ont pas les mêmes conséquences sur les traitements. De manière générale, la phase d'acquisition est la première étape cruciale dans beaucoup de domaines connexes au projet général comme la photogrammétrie. Les acquisitions massives automatiques d'images ne sont pas envisageables si un opérateur humain doit vérifier la qualité de chaque photo. Plusieurs effets indésirables sont possibles : bruit, ensoleillement, flou. Dans le cas particulier de la surveillance d'ouvrage d'art, le problème le plus contraignant est le flou. Le flou peut être causé par un mouvement au moment de la prise de photos ou par un effet hors foyer (out-of-focus) qui peut survenir très facilement en utilisant une focale fixe ou en déplaçant le vecteur d'acquisition. Nous présentons ici plusieurs méthodes génériques aux problèmes de contrôle qualité des images. Puis nous présenterons notre méthode dérivée directement du détecteur FLASH afin de réaliser les calculs directement en une seule passe au moment de l'acquisition.

3.2 Etat de l'art

Dans la littérature, les techniques de contrôle qualité des images sont regroupées sous le mot-clé commun *Image Quality Assemement* (IQA). On peut distinguer 3 catégories :

— Full-Reference QA (FR-QA) : consiste à comparer une image détériorée avec une image de référence.

— Reduce-Reference QA (RR-QA) : utilise une partie de l'information d'une image de référence pour estimer la qualité d'une image.

— No-Reference QA (NR-QA) : utilise aucune image de référence pour réaliser l'estimation de la qualité.

Nous avons focalisé notre recherche sur les méthodes sans apprentissage et sans référence (NR-QA). Les premiers travaux s'inspirent du système visuel humain (HVS),

en modélisant les mécanismes intervenant dans la distinction des images floues et non floues.

3.2.1 Méthodes de NR-QA

Ces techniques se basent principalement sur les contrastes locaux d'une image. Par exemple, le concept de "Just Noticeable Blur" JNB est introduit par [Ferzli09]. Les auteurs se basent uniquement sur des zones locales considérées comme ayant un contour distinct. Selon le contraste, ils mesurent la largeur du contour détecté. Cette mesure sert ensuite à quantifier le flou dans l'image entière. Ce concept a été repris par plusieurs auteurs de différentes manières dans [Narvekar09] en utilisant la fonction de répartition (ou fonction de distribution cumulative).

L'exploitation des contours a également inspiré les travaux de Feichtenhofer dans [Feichtenhofer13]. Les auteurs effectuent une analyse locale du gradient des contours et calculent un "Index de netteté perceptuelle" (PSI).

Toujours basés sur l'HVS, la méthode proposée dans [Bahrami14] analyse également les gradients locaux afin de déterminer la distribution de variation maximale locale (MLV). Cette distribution permet d'évaluer la netteté pour chaque pixel. Elle est calculée sur une fenêtre prédéfinie.

Dans [Li16], les moments discrets de Tchebichef sont utilisés car leurs magnitudes diminuent avec le degré de flou observé sur une image.

Dans [Sieberth16], leur métrique SIEDS est basée sur la différence des canaux de saturation de l'image d'origine et de sa version artificiellement floutée. Les auteurs utilisent leur estimation de flou afin d'identifier dans une base d'images prises par un drone, les images floues et non floues. Ils affirment que la valeur SIEDS dépend de la base de données. Ainsi, il convient de normaliser les valeurs obtenues par rapport à chaque base traitée.

Plusieurs méthodes basées sur une transformation dans le domaine fréquentiel ont été élaborées. Dans les travaux récents, les auteurs de [Gvozden18] opèrent dans le

domaine spatial mais utilisent également la transformée en ondelettes pour calculer une métrique d'estimation de flou. Ils extraient le centile des coefficients d'ondelettes haute fréquence pour estimer la netteté d'une image. [Mavridaki14] calcule le spectre de puissance d'une image et construit un histogramme sur plusieurs fenêtres pré définies. Ensuite, un SVM est utilisé afin de réaliser une classification binaire et non une régression comme on peut le voir en général.

Les réseaux de neurones à convolution sont aussi utilisés, couplés à une fenêtre glissante. Dans [Li17], un réseau peu profond est conçu pour déterminer le flou d'une image. Il s'agit évidemment d'une méthode avec apprentissage.

L'ensemble de ces méthodes a prouvé une efficacité dans des applications diverses, cependant elles ne sont pas adaptées pour déterminer le flou dans une image peu contrastée et avec peu de texture. En effet, elles se basent toutes sur la perception que l'oeil humain a des contours en présence de flou. S'il y a peu de contraste, certains algorithmes sont beaucoup moins robustes. De plus, la rapidité de calcul est un facteur très important dans l'application prévue dans le cadre de la thèse et toutes n'y répondent pas de façon optimale.

3.2.2 Motif Binaire Local (LBP)

Les méthodes précédentes montrent que la notion de flou est liée à une notion de texture très générique où l'on étudie principalement les variations d'intensités. Parmi, les indicateurs pour la caractérisation de texture, on retrouve les motifs binaires locaux (LBP). Aussi pour réaliser la segmentation de zones floues dans une image, des méthodes existantes comme dans [Yi16] ont utilisées les LBP. C'est pourquoi nous allons décrire ici l'opérateur LBP. Nous verrons par la suite que l'opérateur LBP n'est pas sans rapport avec le détecteur FLASH.

Les motifs binaires locaux ou LBP sont une représentation locale de la structure autour d'un point central. Le premier domaine d'application est la reconnaissance faciale. L'opérateur LBP extrait une valeur en analysant P pixels voisins dans un rayon R. En notant g_c l'intensité du point central et g_p, l'intensité du pixel voisin numéro p,

l'opérateur se définit de la façon suivante :

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) \cdot 2^P \tag{3.1}$$

avec

$$s(x) = \begin{cases} 1, & x \geq T \\ 0, & x < T \end{cases} \tag{3.2}$$

Dans la version originale [Ojala02], T est fixé 0 et P et R sont respectivement le nombre de points à comparer et le rayon qui définit la taille de la fenêtre locale à analyser. Généralement, on choisit les paramètres avec $P = 8$ et $R = 1$. L'analyse se fait alors sur les 8 pixels adjacents au pixel central. On obtient alors un code LBP sur 8 bits définissant ainsi 256 structures différentes possibles.

Plusieurs de ces structures sont similaires si l'on considère une invariance en rotation. Il est intéressant de les regrouper par famille. Il existe en effet exactement 36 familles de motifs invariantes en rotation. Mais ce n'est pas forcément suffisant pour avoir une bonne représentation d'une texture. Dans [Ojala02], une autre formule de LBP est définie en considérant l'invariance en rotation et ce qu'on appellera **l'uniformité** d'un motif. Ces motifs sont dits uniformes s'ils ont au maximum 2 transitions spatiales. Par exemple, si l'on note la structure LBP sous forme de code binaire, le code LBP 00000110 et le code 00000000, ont respectivement 2 et 0 transitions spatiales. Ils sont considérés comme des motifs uniformes. Ces derniers nous conduient aux valeurs $LBP^{riu2}_{P,R}$ présentées sur la figure 3.1. En considérant cette caractéristique, nous obtenons une nouvelle expression de l'opérateur LBP comme suit :

$$LBP^{riu2}_{P,R} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & \text{si } U(LBP_{P,R}) \leq 2 \\ P + 1 & \text{sinon} \end{cases} \tag{3.3}$$

avec $U(LBP_{P,R})$ le nombre de transitions spatiales dans le motif. En conséquence, $U(LBP_{P,R}) \leq 2$ représentent les motifs dit uniformes. Les motifs non-uniformes, c'est-à-dire avec un nombre de transitions supérieur à 2 conduiront à une valeur $LBP^{riu2}_{P,R}$ de

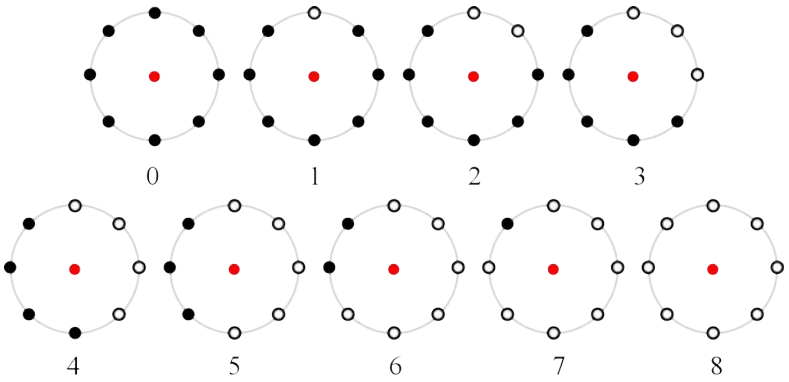


FIGURE 3.1 – Structures uniformes de LBP

9 dans le cas où $P = 8$ de notre exemple.

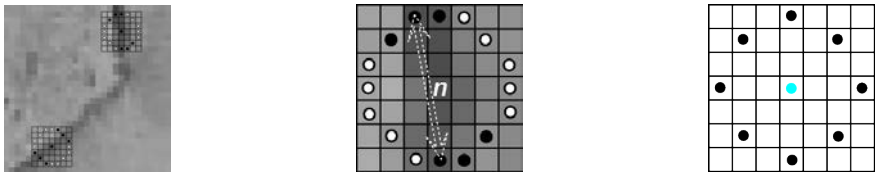
Dans nos calculs, nous avons une meilleure robustesse au bruit en ajustant cette valeur seuil T à une valeur positive (dans nos expériences, T est fixé à 5 par exemple).

3.3 Détection du flou en temps réel

3.3.1 Principe du LBP (dérivé de FLASH)

La détection des fissures permet de réaliser une analyse des pixels dans une zone locale similaire à l'analyse réalisée par l'opérateur LBP. Nous avons 16 pixels autour d'un point central à analyser mais nous avons vu précédemment qu'une optimisation permettait de s'affranchir de l'analyse de tous les pixels. Bien qu'en moyenne 4 pixels soient examinés avec un seuil de détection donné, nous pouvons pousser l'analyse à 8 points minimum sans perdre énormément en temps de calcul pour une application temps réel. On peut alors obtenir une valeur classique de LBP sur 8 bits et avoir 256 configurations à analyser. Cependant, le rayon n'est pas fixé à 1 mais va de 2 à 3. On a alors une approximation de l'opérateur LBP avec des points se situant sur le cercle de Bresenham comme sur la figure 3.2.

Ce nouvel opérateur sera noté LBP_f. Dans la détection de fissures, nous pouvons également calculer dans une même analyse LBP_f. Le lien entre les deux opérateurs est



(a) (b) (c)

FIGURE 3.2 – Analyse FLASH. (a) Exemple de détection de fissure avec l'opérateur FLASH (b) Motif pour la détection FLASH (c) Points analysés dans FLASH pour la construction de LBP_f

important. Comme le montrent les expérimentations, l'algorithme n'est pas robuste au flou. La détection de fissures peut donc être évaluée en fonction de notre métrique de flou dans un processus englobant à la fois l'acquisition et la détection des défauts.

3.3.2 Notre détecteur

De la même façon que les algorithmes existants, nous exploitons les structures fines de l'image pour savoir si elle est floue. Comme dans [Sieberth16], nous basons notre métrique sur le faitqu'une version floutée d'une image déjà floue au départ sera peu différente de l'image floue d'origine, alors que la version floutée d'une image plus nette au départ présentera des différences plus importantes avec la version plus nette de départ. En effet, les transitions nettes sont plus impactées par une opération de floutage que les transitions présentes dans une image déjà floue. Concrètement, on compare donc l'image originale avec sa version artificiellement floutée. Cette dernière est obtenue en appliquant un filtre passe-bas sur l'image d'origine. Par exemple, les plus simples filtres sont les filtres moyenneurs. Les plus utilisés sont les filtres gaussiens. Le choix du filtre n'impacte pas les résultats de l'algorithme.

En pratique, l'histogramme des motifs LBP nous servira d'indicateur déstiné à évaluer le flou et nous utiliserons pour cela la somme normalisée d'une partie de cette histogramme. Cet histogramme a 10 bins, 0 à 8 pour les motifs uniformes et le bin 9 pour les motifs non-uniformes. Le motif 0 n'est pas considéré dans le calcul car une image nette peut avoir des zones complètement homogènes, c'est pourquoi on retrouve un dé-

compte des LBP_f^{riu2} différents de 0 au dénominateur de s sur la formule 3.4. Dans [Yi16], les auteurs montrent que le profil de l'histogramme des LBP change significativement pour les valeurs entre 6 et 9. Nous avons défini notre métrique s comme suit :

$$s = \frac{\mid sLBP - sLBP_{blurred} \mid}{\#(LBP_f^{riu2} \neq 0)} \tag{3.4}$$

avec

$$sLBP = \sum (LBP_f^{riu2} \geq 6)$$

Le score ne dépend pas d'une zone spécifique comme les contours. Le motif '00000000' n'est pas considéré empêchant toute prise en compte de grandes zones homogènes comme un ciel bleu ou un mur peint lisse. La difficulté dans les images peu texturées est de trouver de petits indices qui permettront de déterminer la qualité de l'image. Les motifs non-uniformes ont une large contribution. En effet, il montre la présence d'un grand nombre de transitions dans une région locale. Plus la valeur de notre score est élevée, plus l'image est nette.

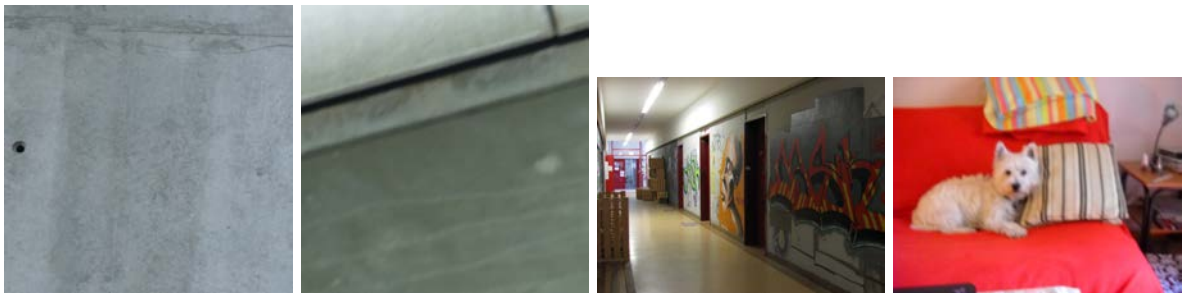
3.3.3 Évaluation

3.3.3.1 Base de données

La plupart des bases de données publiquement disponibles pour l'évaluation sont basées sur des notes de perception d'un être humain. Les sujets sont amenés à noter la qualité des images. Par ces observations, on obtient le score MOS (Mean Opinion Score) pour chaque image. Cependant, dans le contexte d'acquisition automatique, une décision binaire doit être prise : soit on sauvegarde la photographie courante, soit on reprend une nouvelle photographie afin d'avoir une meilleure qualité, suffisante au moins pour le traitement à venir. En conséquence, nous exploitons les bases de données avec des images classifiées floues ou non floues. Le base CERTH [Mavridaki14] est composée de 411 images naturellement floues (dues à une mauvaise mise au point ou un flou de bougé) et 589 images non floues dans le jeu d'évaluation. Nous avons fait le

choix de faire les tests uniquement sur des images naturellement floues.

En plus de cette base, nous avons construit une petite base de données représentant les surfaces en béton. Elle est composée de 210 images au total, avec la moitié destinée à chaque classe. Chaque image a une résolution de 1024x1024 et a été labellisée avec l'aide d'un expert jugeant si la qualité de l'image permettait d'identifier un défaut ou pas. Des exemples des deux bases de données sont montrés sur la figure 3.3.



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

FIGURE 3.3 – Exemples d'images nettes (a) et (c), images floues (b) et (d) respectivement de notre base de d'images et de la base CERTH.

3.3.3.2 Métriques et paramétrages

Les algorithmes sont exécutés sur un ordinateur portable possédant un processeur Intel Core i7 sous windows 10. Les tests sont tous effectués dans Matlab R2017b et en utilisant si cela est possible le code fourni par les auteurs. Ce choix d'une implémentation Matlab qui ne donne pas des temps d'exécution optimaux par rapport à une version C par exemple, est dicté par le besoin de comparer les algorithmes entre eux plutôt que de savoir quel temps minimal ils prendraient avec une implémentation la plus rapide possible. Or, pour la plupart, les sources originales de ces méthodes sont données en version Matlab par leurs auteurs. Les optimisations de Matlab ne sont pas exploitées dans le calcul du LBP. Nous utilisons un filtre passe-bas gaussien avec un noyau de taille 7x7.

Les métriques utilisées pour l'évaluation sont la précision, le rappel et la mesure d'accuracy. La mesure d'accuracy, notée *acc* se calcule de la façon suivante :

$$acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.5}$$

où TP est le nombre d'images floues détectées floues, TN le nombre d'images non floues détectées non floues. La mesure du temps est également essentielle. Elle est réalisée en utilisant les fonctions de Matlab destinées à cet effet. Nous utilisons également la mesure d'aire sous la courbe (*Area Under Curve ou AUC*).

3.3.3.3 Temps de calcul

Le temps de calcul est primordial si l'on veut assurer une acquisition optimale en qualité. Néanmoins, ce sont les rapports relatifs de temps d'exécution qui nous intéressent ici. Le tableau 3.1 montre un récapitulatif des tests effectués avec les algorithmes disponibles. LBP a le second meilleur score. Malgré ce résultat, il faut noter que certaines méthodes utilisent les fonctions prédéfinies de Matlab qui sont optimisées. SIEDS a le meilleur temps mais n'est pas robuste en terme d'accuracy (ou exactitude). ARISMC a pris plus de 2 jours de traitement. MLV a le pire temps de traitement après ARISMC mais obtient de bons résultats au niveau de l'exactitude sur la base de données CERTH. De même PSI obtient une bonne performance, aussi bien sur notre base de données que en temps de calcul, en comparaison avec notre métrique.

Bien que nous devions effectuer nos calculs sur la version originale et la version floutée de l'image, notre méthode possède un avantage certain : c'est la possibilité d'effectuer le calcul uniquement sur la version floutée de l'image afin d'obtenir l'indice de flou, si nous avons préalablement effectué une détection de fissures. Ainsi, en suivant le processus complet d'acquisition, nous pouvons approximativement diviser le temps de traitement par un facteur 2 sur les temps affichés sur le tableau 3.1.

L'ensemble des méthodes mentionnées ici sont rappelées dans la section état de l'art du chapitre.

Méthodes	Temps (s)
Notre méthode	536
BISHARP [Gvozden18]	1287
MLV [Bahrami14]	1597
ARISMC[Gu15]	+40000
PSI [Feichtenhofer13]	632
SIEDS [Sieberth16]	340
BRISQUE [Mittal12]	1141

TABLE 3.1 – Temps de calcul pour traiter l'ensemble de la base de données CERTH. En rouge le meilleur score, en bleu le second meilleur score.

Notre jeu de données								
Métriques	Contribution	BISHARP	MLV	ARISMC	PSI	SIEDS	BRISQUE	LBP
acc	.947	.928	.942	.914	.971	.547	.533	.781
rappel	.943	.941	.942	.906	.962	.528	.525	.743
precision	.952	.914	.942	.923	.981	.904	.676	.857
AUC^{1}	.990	.977	.981	.947	.985	.475	.508	.809
$AUC^{0.25}$	.978	.819	.870	.004	.160	.460	.959	.541
CERTH dataset								
Métriques	Contribution	BISHARP	MLV	ARISMC	PSI	SIEDS	BRISQUE	LBP
acc	.794	.749	.760	.530	.677	.750	.652	.752
rappel	.823	.695	.742	.639	.669	.724	.656	.823
precision	.635	.693	.637	.463	.893	.632	.857	.736
AUC^{1}	.819	.809	.828	.558	.600	.662	.662	.790
$AUC^{0.25}$	.861	.723	.777	.348	.144	.790	.695	.675

TABLE 3.2 – Résultats sur deux bases de données. En rouge, les meilleurs résultats, en bleu, les seconds.

3.3.3.4 Efficacité

Nous présentons les résultats sur les 2 bases de données décrites ci-dessus dans le tableau 3.2. Notre méthode a été comparée à 2 autres basées sur des mécanismes d'apprentissage, BRISQUE et une basée sur LBP [Yue17]. Pour cela, les modèles disponibles dans Matlab ont été utilisés pour BRISQUE. Un SVM a été entraîné sur la base d'entraînement de CERTH. Sur notre base de données d'images avec peu de textures sur les images, beaucoup d'algorithmes existants obtiennent des scores similaires. Notre métrique présente le meilleur score en terme d'aire sous la courbe (AUC). Les méthodes basées sur de l'apprentissage ne sont pas très efficaces. Il s'agit de plus d'une contrainte

qui impose de connaître le type d'images que l'on va traiter. Dans notre cas, plusieurs types de cas de figures peuvent se présenter dans la surveillance d'ouvrages d'arts, ce qui serait un inconvénient pour ces méthodes.

Globalement, nous obtenons de très bons scores (second meilleur ou meilleur) sur les deux ensembles de données. Notre méthode a donc une capacité de flexibilité car elle obtient aussi de bons résultats sur des images plus diverses que celles de notre application spécifique. Elle n'est donc pas uniquement destinée à traiter des images de surfaces en béton.

3.4 Evaluation pour la détection de fissures

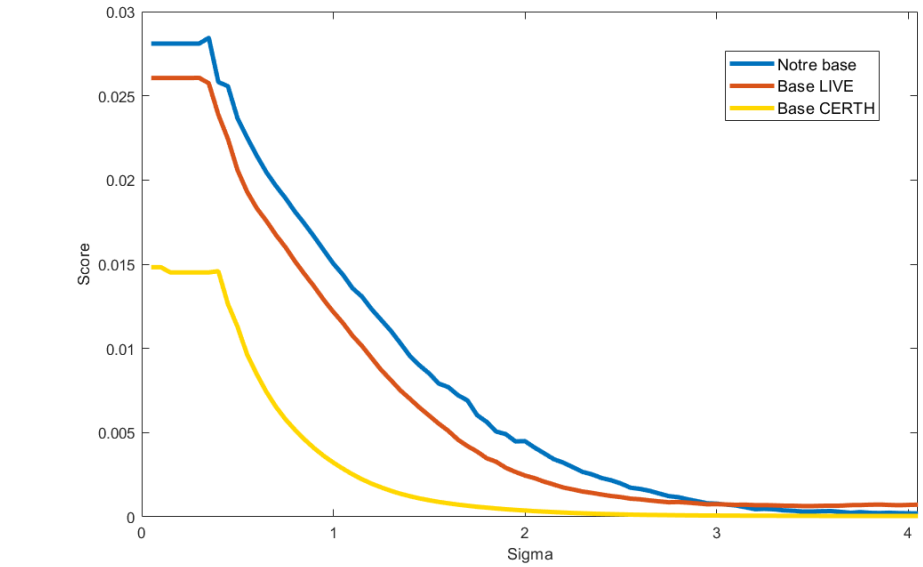
3.4.1 Protocole

Nous rappelons que l'objectif est de détecter des fissures de 0,1 mm d'ouverture dans des images avec une résolution de 0,5 mm par pixel. Ainsi, les fissures sont sous-pixels dans l'image. Il est important d'établir un lien entre la capacité de détection de FLASH et le flou dans l'image.

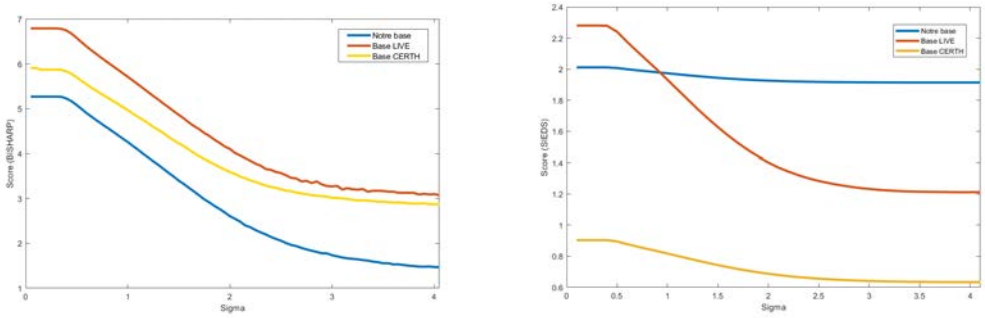
Pour évaluer la robustesse au flou et le seuil de détection permis pour FLASH, nous avons testé le seuil critique de détection en utilisant des filtrages gaussiens différents. Le seuil critique est fixé de la façon suivante : il s'agit du moment où la détection FLASH perd plus de 50% de son taux de détection initial.

Plusieurs tailles de fissures, estimées en nombre de pixels, ont été testées. Pour chaque taille, on applique des filtres gaussiens pour simuler le flou dû à une mauvaise mise au point. Ensuite, nous évaluons le taux de détection de notre algorithme FLASH.

Quelques images de notre base de données ont été utilisées. Elles sont moins nombreuses car nous n'avons pas la connaissance de la taille réelle des fissures. Nous avons une image de référence pour effectuer les tests.



(a)



(b) (c)

FIGURE 3.4 – Analyse de la détection de flou. (a) Notre méthode (b) BISHARP. (c) SIEDS

3.4.2 Résultats

Tout d'abord, nous présentons la mesure obtenue par notre méthode en fonction du flou de l'image dans la figure 3.4. Les tests ont été réalisés sur trois bases de données : la base LIVE, la base CERTH et notre base d'images de béton. La base LIVE [Sheikh06] est une base de référence dans le domaine de l'IQA. Elle est composée d'images de scènes diverses qui sont artificiellement floutées pour évaluer les métriques d'estimation de

flou. Nous avons appliqué un flou gaussien avec un noyau de taille 11x11 et une valeur de sigma variant de 0,05 à 5,05 pour un pas de 0,05. Nous obtenons ainsi une évolution de notre mesure sur 100 valeurs.

Nous pouvons observer quelques différences entre les images issues de mur en béton et celles issues de scènes diverses. En effet, dans la base LIVE (base 2), la faible profondeur de champ de certaines photographies ont pour effet de produire un flou artistique en fond. Notre mesure étant calculée sur l'ensemble de l'image, la valeur calculée se retrouve légèrement plus basse que la valeur de nos images de surfaces de béton qui ne présentent aucun flou du même genre.

Il est important de noter que nous observons le même profil de courbe selon le type d'images ainsi que la même dynamique de valeurs. Cela montre la possible généralisation de notre mesure de flou.

On note que la méthode BISHARP n'est pas adaptée à des images avec peu de textures. La dynamique des valeurs obtenues pour notre base est inférieure à celle des autres bases. De plus, la courbe décroît plus doucement que pour notre méthode. Notre objectif étant de déterminer si une image est floue ou non, il est important d'obtenir une courbe qui décroît très rapidement lorsque l'on note la présence de flou.

La méthode SIEDS est la plus rapide et est destinée à détecter des images floues dans une base d'images de drone. La courbe (c) de la figure 3.4 montre que la présence de flou dans une image texturée entraîne un score très faible (base 3). La méthode SIEDS n'est pas adaptée aux images peu texturées (base 1). En effet, la courbe est quasiment constante rendant impossible la distinction entre images floues et non floues.

La détection de fissures

Afin d'avoir une meilleure précision pour déterminer le seuil critique de détection, nous utilisons le nombre de pixels fissures détectés pour calculer le pourcentage de détection. Initialement, la détection est à 100% de détection. Plus l'image est floutée, plus le pourcentage de détection diminue. Pour réaliser cela, nous avons déterminé des

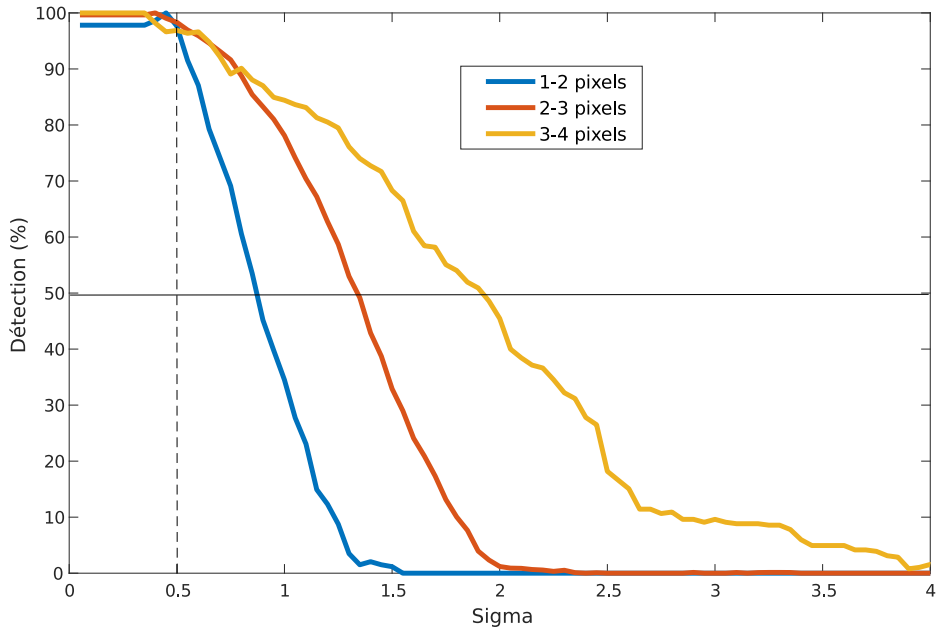


FIGURE 3.5 – Analyse de la robustesse de notre détection face au flou selon la taille de la fissure en pixels et le flou de l'image.

zones de présence de la fissure et compté le nombre de pixels détectés comme appartenant à une fissure.

Sur la figure 3.5, nous notons sans surprise que la détection de fissures est d'autant plus robuste que la largeur de la fissure est grande. Nous remarquons également que pour nos images de tests, un pic de la détection est observé juste avant que la courbe ne décroisse pour une valeur de sigma avoisinant $0,5$. Cela montre que la présence de bruit dans l'image non floue peut perturber légèrement la détection de fissures et que l'application d'un léger flou gaussien peut entraîner une amélioration de la détection des fissures par FLASH.

Les autres méthodes de détection de fissures sont robustes au flou. Cependant, il est important de noter que nous avons besoin d'image de qualité aussi bien pour la détection des défauts que pour l'élaboration de modèles 3D texturés.

3.5 Conclusion

La détermination de la qualité d'une image est une étape essentielle dans le processus d'acquisition des données. Elle permet de savoir s'il faut refaire une acquisition et si la détection de fissures sera pertinente. En effet, la détection de fissures étant sensible au flou, il faut pouvoir dire très rapidement si le traitement est possible. Nous avons donc développé une méthode reprenant les calculs réalisés dans la détection de fissures afin qu'elle soit très rapide. Bien que la métrique soit inspirée de méthodes existantes, l'application que l'on en fait ainsi que sa rapidité de calcul en font une méthode pertinente. De plus, les résultats montrent une polyvalence intéressante de notre méthode puisqu'elle donne des résultats remarquables avec d'autres types d'images, pour d'autres applications, là où d'autres méthodes de l'état de l'art n'arrivent pas à donner de bons résultats sur nos images faiblement ou très faiblement contrastées.

4

Appariement d'images selon FLASH

4.1 Introduction

L'appariement d'images est une étape primordiale dans plusieurs domaines d'applications de la vision par ordinateur. Il assure la faisabilité de certaines étapes de reconnaissance d'objets ou d'analyse de documents en permettant une mise en relation des éléments communs à deux ou plusieurs images. Dans le cadre de notre application, ce n'est pas la reconnaissance d'objets qui motive le recours à l'appariement d'images mais une application très spécifique : le SLAM (Simultaneous Localisation and Mapping). L'objectif est ici de reconstruire un panorama plus étendu de la scène à partir de différentes prises de vue qui constitueront les éléments d'une mosaïque beaucoup plus étendue. L'intérêt de cette reconstruction est double. D'une part elle permettra de suivre les défauts détectés sur deux (ou plus) images dans le cas de défauts qui s'étendent sur des distances ou des surfaces importantes ou tout simplement si le défaut, même petit se trouve à cheval entre deux images. Mais cette reconstruction permettra aussi de situer la position du drone de façon plus précise sur l'ouvrage d'identifier des formes ou structures particulières qui apparaissent régulièrement sur ce type d'ouvrages comme les jonctions entre des plaques de béton ou les trous circulaires d'évacuation d'eau.

Historiquement, les résultats les plus intéressants, d'appariements d'images ont commencé à être obtenus avec l'utilisation des points d'intérêts. Cependant, avec l'essor des réseaux de neurones, autour des années 2000, les points d'intérêts sont moins utilisés pour certaines taches. Ils sont toutefois toujours à la base de nombreux algorithmes. Outre l'appariement d'images, les points d'intérêts étaient notamment utilisés pour la reconnaissance d'objets ou de formes, l'indexation des images et la caractérisation de contenu. Pour ce qui nous concerne, nous poursuivrons cette approche à base de points d'intérêts pour réaliser les appariements entre images, car d'après nos tests, ils se sont montrés plus performants sur nos images sans texture et sans contraste que les approches à base de réseaux de neurones que nous avons pu envisager. Cela s'explique sans doute par la difficulté de faire apprendre à un réseau de neurones la spécificité d'une zone très peu contrastée pour l'apparier à une autre zone similaire sur une autre image. En revanche, les points d'intérêt permettent justement de se focaliser sur des

caractéristiques précises et en nombre limité, comme nous allons le voir, ce qui permet un appariement par similarités locales très efficace.

Nous nous intéressons ici à des appariements dans des environnements sans texture particulière, très peu contrastés, où rien ne ressemble plus à une zone de mur de béton homogène qu'une autre zone de mur de béton homogène.

Dans ce chapitre, nous allons aborder le problème de l'appariement d'images, ou de mise en correspondance. Cet appariement peut se réaliser sur plusieurs images, mais on peut le décomposer et le simplifier en traitant l'appariement entre deux images. Il s'agit de déterminer si ces deux images ont une partie commune ou non, et si oui quelles sont les transformations à appliquer sur la première image pour l'amener en correspondance (partielle le plus souvent, mais on peut envisager aussi une inclusion complète) avec la deuxième image. Ces transformations sont la translation, la rotation et le changement d'échelle dans les cas les moins compliqués. On peut envisager aussi de traiter les situations pour lesquelles une transformation de type perspective est aussi nécessaire. Dans notre cas, et compte tenu des contraintes imposées comme pré-requis aux conditions d'acquisition des images de notre problème de contrôle de qualité des ouvrages d'art, nous pouvons supposer que ces transformations perspectives ne seront pas à envisager puisque toutes les images sont prises perpendiculairement à la surface à contrôler. Par ailleurs, les changements d'échelle sont inévitables, mais restent dans une plage relativement réduite de facteurs d'échelle (on pourrait l'estimer entre 0,75 et 1,3 environ). C'est donc avec ces hypothèses préliminaires que nous nous intéresserons à ce problème de l'appariement entre deux images.

Nous définissons un point d'intérêt de la façon suivante : il s'agit d'une région spécifique et peu étendue et visuellement informative organisée autour d'un point qui l'identifie. Deux phases sont à distinguer mais sont indispensables à la construction d'un point d'intérêt : La détection et la description (ou extraction d'une signature spécifique à partir de caractéristiques locales). Dans la partie qui suit, nous présenterons d'abord les principales méthodes existantes de détection et de caractérisation de points d'intérêt et les raisons de leur inefficacité au contexte d'inspection d'ouvrages d'art

puis nous proposerons notre solution basée sur les calculs de FLASH. Nous développons un algorithme de correspondance efficace qui permet d'obtenir directement les transformations à appliquer entre les deux images à apparier, c'est-à-dire, la translation la rotation et le facteur de changement d'échelle.

4.2 État de l'art

Un appariement d'images est le plus généralement réalisé à l'aide d'un détecteur qui capture un point/une région qui sera le plus informatif possible, et d'un descripteur qui cherchera à décrire de manière unique ce point ou cette région. Ces deux étapes de *détection* et de *description* peuvent être définies de façon indépendante et associées par choix, ou être définies ensemble dans un même processus complet. Ce sont ces détecteurs / descripteurs que nous allons détailler dans la suite. On distingue principalement 3 types d'algorithmes :

— ceux basés sur les points
— ceux basés sur des lignes. Ils sont plus précis car un ensemble de lignes est plus discriminant qu'un ensemble de points.
— ceux basés sur des régions. moins précis et beaucoup plus couteux en temps de calcul.

On évalue la robustesse d'un appariement entre deux images selon la rotation, le changement d'échelle et le changement de point de vue que les images ont pu subir. Ces changements peuvent s'additionner rendant la mise en correspondance très difficile. Les algorithmes dans la littérature sont évalués selon ces critères et des bases de données publiques sont disponibles pour les protocoles d'évaluation.

4.2.1 Les détecteurs

Avant de pouvoir décrire un point d'intérêt, il faut pouvoir localiser l'endroit. Un détecteur de points sera performant s'il assure les critères suivants ([Mikolajczyk05]) :

— Robustesse : la détection doit être la même quelle que soit la transformation de l'image. Résistance aux changements d'échelle, rotations, bruit...
— Répétabilité : un point caractéristique d'un objet/scène doit ressortir à la même localisation sur l'objet ou la scène dans différentes images et sous différentes conditions de visualisation, après différents changements (angle de vue, luminosité, compression -ou perte de qualité, occlusion, etc.) .
— Précision : la localisation de la détection des points d'intérêt est assez précise.
— Généralité : La détection peut se faire dans différents domaines d'application.
— Efficacité : La détection est assez rapide et peut être réalisée avec des contraintes de temps fortes.
— Quantité : les points d'intérêt détectés sont suffisamment nombreux sur une image pour être représentatifs de toutes les caractéristiques que l'on peut trouver dans cette même image.

Beaucoup de détecteurs sont décrits dans la littérature. Nous présentons ici les plus répandus et les plus utilisés.

Détecteur de coins de HARRIS

Les coins sont des zones d'une image qui présentent une forte variation en intensité. Ils sont également robustes aux déformations d'une image. Les points de HARRIS sont construits pour correspondre à des coins basés sur l'intensité lumineuse des pixels. [Harris88] propose un détecteur à partir de l'expression d'auto-corrélation des variations d'intensité suivantes :

$$M(x,y) = \sum_{(u,v)} w(u,v) \cdot \begin{pmatrix} I_x^2(x,y) & I_x I_y(x,y) \\ I_x I_y(x,y) & I_y^2(x,y) \end{pmatrix} \tag{4.1}$$

avec I_x et I_y respectivement les dérivés locales en x et y, $w(u,v)$ est une pondération sur la fenêtre (u, v). L'étude des valeurs propres de la matrice M permet de déterminer si un point est un coin, une région homogène ou un contour. Un dernier critère est calculé à partir de M permettant de prendre la décision sur le type de point trouvé.

Ce détecteur a été très utilisé car il est simple à implémenter et permet d'avoir toutes les caractéristiques des points d'intérêt trouvés.

SUSAN (Smallest Univalue Segment Assimilating Nucleus) et FAST

Ces détecteurs ont déjà été évoqués dans le chapitre 2. On rappelle que ces détecteurs reposent sur l'analyse locale de la structure autour d'un point central. Cette analyse est réalisée avec les intensités lumineuses. D'autres détecteurs se sont inspirés de FAST comme Oriented Fast[Rublee11] ou AGAST[Mair10]. Retenons que l'implémentation de FLASH est inspirée de l'implémentation de FAST. Les 2 algorithmes sont très efficaces en temps de calcul et peuvent donc être utilisés sur des applications temps réel.

Détecteur SIFT

Le détecteur SIFT a été introduit dans [Lowe04] en même temps que son descripteur décrit dans la section suivante. Cette méthode est assez consommatrice en temps de calcul mais reste performante car invariante aux changements d'échelle. La différence de gaussienne (DoG) est réalisée en soustrayant une image à sa version convoluée par une gaussienne G. La fonction DoG est définie par :

$$DoG(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma) \cdot I(x, y) \tag{4.2}$$

avec k le facteur entre les deux gaussiennes, définissant ainsi la différence de finesse des points d'intérêt observés. Ce calcul est réalisé dans un espace de plusieurs octaves. On recherche les extrema, correspondant alors aux points d'intérêt recherchés, dans l'espace échelle construit. La différence de gaussiennes peut être vu comme une approximation du laplacien de gaussiennes. Plusieurs variantes du détecteur SIFT ont été proposées, comme le détecteur SURF [Bay08] qui propose d'améliorer le temps de calcul en approximant le filtre gaussien par un filtre rectangulaire.

LSD

LSD [Gioi12] est très efficace et permet une détection sans paramétrage. La détection de ligne est décrite dans la section 2.2.

La transformée de Hough

Technique très ancienne et utilisée massivement par la communauté scientifique, elle permet de décomplexifier le problème de reconnaissance de formes simples et de transformer en un problème plus simple de recherche de maxima. C'est le transformée de Hough [Duda72]. On peut l'appliquer à la reconnaissance de lignes et également de cercles. L'espace de Hough est un espace à deux dimensions dans lequel les paramètres d'une droite passant par un point sont représentés par un point. L'ensemble des droites passant par ce même point constitue donc une courbe dans l'espace de Hough. Intuitivement, si 2 points se situent sur la même droite alors leur courbe se couperont en un point dans l'espace de Hough. Des paramètres supplémentaires viennent s'ajouter comme la distance minimale entre 2 points ou le nombre de points qu'il faut pour considérer une droite.

4.2.2 Descripteurs

Un descripteur a pour rôle de caractériser la discriminabilité ou distinctivité d'une zone identifiée par un point : cela se construit à partir d'informations discriminantes de la zone autour du point d'intérêt. Dans une même image, les points d'intérêt doivent pouvoir se distinguer les uns des autres par leur signature spécifique.

Là encore, plusieurs descripteurs de points d'intérêt sont abordés. Nous ferons un focus sur les descripteurs pouvant s'utiliser en temps réel ou proche temps réel.

Certains descripteurs sont indissociables du détecteur utilisé (un détecteur de lignes avec un descripteur de ligne).

4.2.2.1 Descripteurs non-binaires

Descripteur SIFT

Un des plus connus et également très gourmands en ressource est le SIFT et son adaptation/amélioration SURF. [Lowe04] Il s'agit de méthodes avec des descripteurs flottants, c'est à dire n'utilisant pas des chaînes binaires. Autour d'un point d'intérêt, SIFT calcule un histogramme des orientations du gradient dans une région subdivisée en $n \times n$ sous-régions. Ces orientations sont pondérés par l'amplitude du gradient. Dans [Lowe04], les auteurs choisissent $n = 4$ et un histogramme à 8 orientations. Le vecteur descripteur est alors un vecteur de 128 entiers. Plusieurs étapes intermédiaires permettent d'obtenir un vecteur invariant aux changements d'échelle, au rotation et une certaine robustesse aux changements d'illumination (par exemple, rotation des histogrammes ou une normalisation du vecteur).

Ce descripteur est parmi les plus emblématiques dans le domaine. Sa robustesse en fait un descripteur très utilisé aujourd'hui. Son défaut est principalement son temps de calcul. Cependant, plusieurs améliorations algorithmiques ont été proposées et les avancés technologiques (par exemple le GPU) réduisent un peu l'inconvénient du temps de calcul.

Shape Context

Dans [Belongie02], les auteurs considèrent la forme des objets par leurs contours. Ces derniers sont constitués d'un ensemble de points obtenus à partir d'algorithmes bas-niveau tels que Canny. L'analyse de la distribution des points autour d'un point d'intérêt permet d'avoir une information descriptive. La zone locale analysée est divisée en plusieurs secteurs angulaires réguliers ou non. L'invariance au changement d'échelle est assurée par une normalisation des distances des points voisins du point d'intérêt. L'invariance en rotation est assurée par l'angle de la tangente du contour. Le descripteur est constitué par un histogramme des secteurs avec le nombre de points présents. L'algorithme de mise en correspondance calcule une distance de χ^2 entre les histogrammes en prenant en compte une mesure par rapport à l'orientation.

Cette méthode présente une robustesse à certaines déformations géométriques mais reste performante principalement sur des formes bien définies comme les logos.

4.2.2.2 Descripteurs binaires

Les descripteurs binaires utilisent des chaines de bits afin d'effectuer une mesure de distance de Hamming plutôt qu'une distance euclidienne ou de Mahalanobis. L'intensité entre 2 pixels est comparée autour d'un pixel potentiellement point d'intérêt. Cette comparaison donne une description binaire d'un point d'intérêt et permet d'atteindre de très hautes performances en temps de calcul lors de la phase de calcul de distance entre deux descripteurs.

BRIEF

[Calonder12] propose de comparer de manière aléatoire des pixels dans une région locale autour du pixel examiné. Dans un patch de taille NxN, un nombre fixe de tests est réalisé. La localisation de ces tests est définie dès le début par une distribution gaussienne isotropique.

ORB

Le principe de BRIEF est repris dans [Rublee11]. Les auteurs améliorent la localisation des pixels à tester par une méthode d'apprentissage. Cette dernière vise à maximiser la variance des paires en minimisant leur corrélation. La fenêtre de la région autour d'un point d'intérêt est de 31x31 pixels. La simplicité de calcul de ORB et sa robustesse en font un des descripteurs les plus appréciés dans les applications temps réel aujourd'hui.

BRISK

Dans la même idée de comparer les intensités de pixels, BRISK [Leutenegger11] propose une autre version pour construire le descripteur binaire. Autour du point central sont définis plusieurs points équitablement répartis. On obtient ainsi un motif d'échantillonage adapté localement. Pour chaque point, un rayon de lissage correspondant à la

déviation standard du filtre gaussien est défini. Ce rayon est plus grand pour les points les plus éloignés du point central. Dans la version originale, 512 paires de points sont testées afin de construire le descripteur. Enfin, le calcul de l'orientation diffère de ORB car il est basé non sur l'intensité des pixels voisins mais sur le gradient des paires de points les plus éloignées du point clé central.

LBP

Les LBP décrits dans la section 3.2.2 sont aussi utilisés comme descripteurs. Plusieurs variantes ont été élaborées dans la littérature.

4.2.3 Mise en correspondance

L'objectif d'un appariement est de reconnaitre des éléments communs à deux images (ou plus) et d'estimer alors la transformation géométrique entre ces images.

Après avoir détecté des points d'intérêt et calculé leur description, il faut les identifier dans des images différentes.

Distances

De manière générale, pour des descripteurs flottants (vecteur de réels), on utilise une simple distance L2. Pour les descripteurs binaires, la distance de Hamming sera utilisée. Pour les détecteurs de lignes, les algorithmes sont parfois adaptés en fonction du descripteur associé. Par exemple, les auteurs dans [Zhang13] et [Fan12] construisent leur propre mesure de similarité pour faire correspondre les attributs de segments détectés dans une image.

Ces comparaisons conduisent à des confusions si deux points d'intérêt représentant des éléments différents sont très proches (distance des descripteurs).

4.2.4 Problème de confusion

Après les 3 étapes précédentes de détection de point d'intérêt, de description et enfin de mise en correspondance des points d'intérêt, des confusions peuvent avoir lieu.

La mise en correspondance conduit alors à de mauvaises estimations de la transformation entre les deux images à apparier. Plusieurs techniques existent déjà dans la littérature pour faire face à cela. RANSAC [Fischler81] est une méthode d'estimation d'un modèle mathématique qui vise à ne pas prendre en compte les données trop particulières (Ici, ces données correspondent aux mauvaises correspondances) mais à favoriser un résultat plus "fréquent". La méthode permet donc de privilégier les bonnes correspondances en se basant sur la cohérence du modèle calculé. Malheureusement, il s'agit d'une méthode itérative gourmande en temps de calcul.

4.3 Contribution

4.3.1 Détecteur FLASH étendu

Pour cette phase d'appariement, nous proposons d'exploiter les calculs et les résultats dont nous disposons déjà à ce stade de l'analyse de nos images, à savoir les points FLASH extraits pour la détection des fissures. Comme nous l'avons expliqué dans la section 2.4 précédente, nous avons à notre disposition trois types de points FLASH : les points de type *micro-lignes*, les points de type *coins* et les points de type *bulles*. Lors de la détection des fissures, seuls les deux premiers types de points ont été utilisés. Pour cette phase d'appariement d'images, nous allons utiliser les trois types de points. Et même si, en effet, les micro-lignes et les coins sont les plus informatifs et discriminants puisqu'ils contiennent une information importante de direction, le dernier type de points *bulles* n'en reste pas moins intéressant en augmentant le nombre de points FLASH pouvant contribuer à la définition de signatures locales.

Une fois la phase de détection réalisée, plusieurs points ont été extraits avec leur orientation et leur score. Ces deux dernières caractéristiques vont être primordiales dans l'élaboration du descripteur décrit ci-dessous. Le descripteur et l'algorithme d'appariement ont été conçus conjointement pour être à la fois le plus précis possible mais aussi le plus rapide possible.

4.3.2 Descripteur FLASH

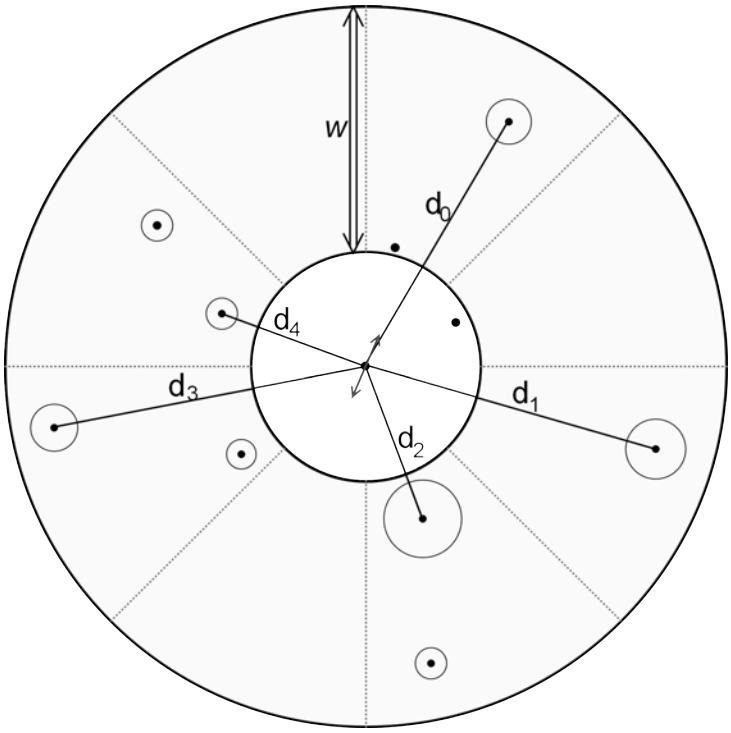


FIGURE 4.1 – Descripteur FLASH. La zone est divisée en 8 secteurs. Le score est représenté par le rayon du cercle autour des points. w est la largeur de l'anneau. 5 points sont retenus pour ce descripteur.

Nous proposons de décrire un point clé en fonction de la distribution des points voisins. Avant d'établir la description d'un point, nous normalisons les descripteurs en fonction de l'orientation du point considéré. Nous réalisons ainsi un descripteur de forme. La robustesse de l'orientation étant importante, nous utilisons uniquement les micro-lignes. Afin de maintenir un temps de traitement raisonnable proche du temps réel, nous décrivons un point à l'aide d'une chaine de bits. Le voisinage de chaque micro ligne est exploré. Dans ce but, un disque est défini autour de la micro ligne avec un rayon R. Ce disque est divisé en plusieurs secteurs angulaires. Chaque bit représente un secteur angulaire. L'idée est que dans un secteur angulaire, la présence d'un autre point clé détermine le bit correspondant (le bit est alors élevé à 1). Si plusieurs points

clés sont présents dans un même secteur angulaire, le point clé ayant le plus haut score est sélectionné. En cas d'égalité, c'est le point le plus proche du point central qui sera choisi. Par exemple, pour le point considéré sur la figure 4.1, la chaîne binaire associée sera 10110110 pour un angle de 45 degrés dans chaque secteur.

La chaîne binaire est une première information sur l'environnement autour d'une micro ligne. Cependant, l'information n'est pas suffisante pour discriminer les points. Elle permet surtout de faire une première sélection dans l'algorithme d'appariement. Afin de discriminer les points clés, les distances entre les points clés et la micro ligne centrale considérée sont sauvegardées. On note d_i la distance associée au i^{eme} bit de la chaîne binaire représentant le secteur i.

4.3.3 Algorithme d'appariement

Chaque description est comparée à toutes les autres de l'image. Des simplifications simples peuvent être effectuées si on connaît la nature d'un déplacement, la zone que l'on a ciblée etc. On note FD_A un descripteur de l'image A et FD_B un descripteur de l'image B. On note alors $FD_A.bs$ et $FD_A.d_i$ respectivement la chaîne binaire et la distance enregistrée dans le secteur i, associées à un descripteur de l'image A. Un score de dissimilarité est calculé et sera noté $S(FD_A, FD_B)$ ou plus simplement S. Le calcul de ce score est décomposé en 3 étapes et est totalement résolu à l'équation 4.7.

Etape 1

Une première comparaison est réalisée à l'aide des chaînes binaires. Les opérateurs classiques XOR et AND sont utilisés. Le XOR correspond ici à la distance de Hamming entre les deux chaines binaires. Ce calcul est très rapide et permet d'éliminer très rapidement les points clés n'ayant pas les mêmes formes avoisinantes. Le détecteur n'étant pas infaillible, une tolérance est établie. La condition pour passer à la prochaine étape est la suivante :

$$\frac{XOR(FD_A.bs, FD_B.bs)}{AND(FD_A.bs, FD_B.bs)} < 0.5 \quad \text{avec} \quad AND(FD_A.bs, FD_B.bs) >= 6 \tag{4.3}$$

Par conséquent, pour continuer l'évaluation du score de dissimilarité, deux descripteurs doivent avoir en commun plus de la moitié des bits "1" en commun. Afin d'améliorer la robustesse, nous fixons à 6 le nombre de points clés minimum en commun dans les secteurs angulaires. En effet, il faut au minimum 3 points pour retrouver la transformation affine liant les 2 images. Avec une erreur de moins de la moitié des secteurs remplies (voir équation 4.3), on obtient ainsi assez de points pour calculer la transformation à partir d'une unique correspondance correcte.
En pratique, nous subdivisions la zone avec 90 secteurs, soit un secteur angulaire de 4 degrés.

Étape 2

Une fois la forme générale vérifiée, il faut vérifier la cohérence en distance. Nous rappelons que la distance existe uniquement s'il existe un point dans le secteur angulaire. Nous comparons donc les secteurs angulaires où l'on retrouve la présence d'un point clé (correspondant simplement à l'opération AND). Nous calculons le ratio α qui représente directement l'échelle entre les images. Notons $setCED$ l'ensemble des points en commun dans les secteurs angulaires.

$$\alpha = \frac{\sum_{i \in setCED} (FD_A.d_i \cdot FD_B.d_i)}{\sum_{i \in setCED} (FD_A.d_i)^2} \tag{4.4}$$

Ce ratio est calculé dans le cas idéal et a été obtenu à partir de la recherche de l'erreur quadratique minimale entre les points. Classiquement, on cherche le minimum de la fonction correspondant à l'erreur :

$$\begin{cases} Err_{distance} = \sum_i (FD_B.d_i - \alpha \cdot FD_A.d_i)^2 \\ et \\ \frac{dErr_{distance}}{d\alpha} = \sum_i 2 \cdot FD_A.d_i (FD_B.d_i - \alpha \cdot FD_A.d_i) \end{cases} \tag{4.5}$$

Il s'agit d'une fonction quadratique dont le minimum est obtenu lorsque sa dérivée est nulle. La résolution de cette équation donne alors l'expression du ratio optimal 4.4.

Etape 3

Nous pouvons ensuite effectuer le calcul final du score de dissimilarité en évaluant la variation moyenne du ratio (ou erreur de stabilité) α_{error}

$$\alpha_{error} = \frac{\sum_{i \in setCED} (FD_B.d_i - \alpha \cdot FD_A.d_i)}{AND(FD_A.bs, FD_B.bs)} \tag{4.6}$$

On peut finalement en déduire le score de dissimilarité comme suit :

$$S = \frac{XOR(FD_A.bs, FD_B.bs)}{AND(FD_A.bs, FD_B.bs)} \cdot \alpha_{error} \tag{4.7}$$

Le score le plus bas correspond à la meilleure correspondance possible que nous pouvons avoir. Pour augmenter le nombre de correspondances correctes entre les points FLASH, nous validons un point de l'image B comme étant la meilleure correspondance pour le point actuel de l'image A ; seul son score est le meilleur (le plus bas) parmi tous les points de l'image B *et* le deuxième meilleur score est suffisamment plus mauvais que le meilleur. En pratique, nous nous attendons à ce que le rapport entre les scores du meilleur et du deuxième meilleur soit inférieur à $0{,}7$.

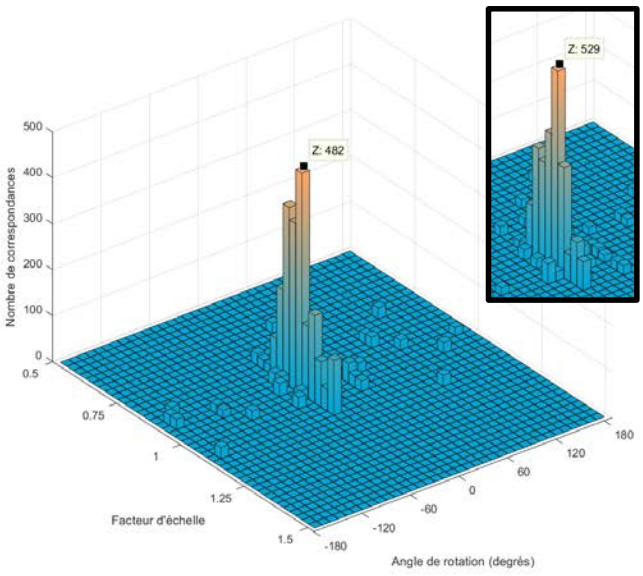
Les calculs d'erreur nécessitent plus de ressources mais augmentent la précision de l'algorithme de correspondance. Si nous avons une contrainte de temps difficile, dans un système intégré par exemple, une version de score dégradé, S_{degrad} peut être obtenue en *supprimant le facteur* α_{error} *de* S.

$$S_{degrad} = \frac{XOR(FD_A.bs, FD_B.bs)}{AND(FD_A.bs, FD_B.bs)} \tag{4.8}$$

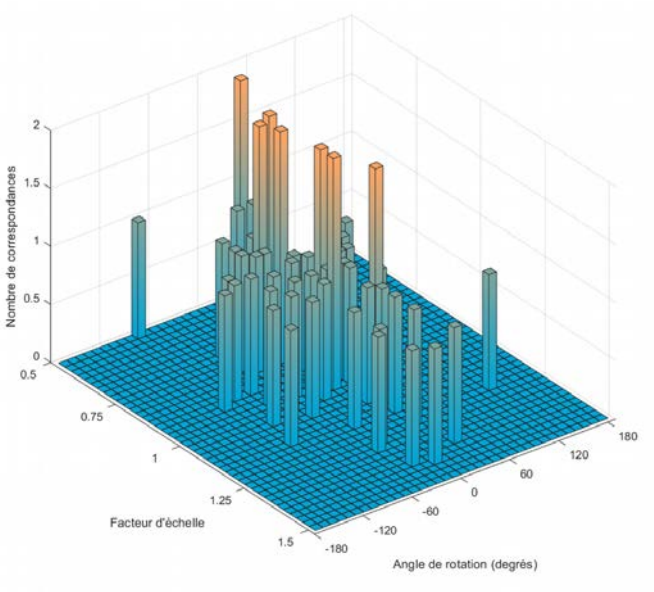
Ainsi, des ressources sont libérées parce que les distances ne sont pas utilisées. Outre l'amélioration du coût de calcul, nous pouvons limiter le nombre de bits à "1" afin d'améliorer le caractère distinctif.

4.3.4 Notre accumulateur

Après les trois étapes précédentes, nous obtenons un ensemble de correspondances entre les descripteurs des images A et B. Toutes les correspondances trouvées ne sont pas nécessairement correctes : il y a alors confusion. Pour vérifier la cohérence de l'appariement et ainsi limiter la confusion, il faut pouvoir détecter les valeurs aberrantes. Nous proposons un nouvel accumulateur de modèle adapté à notre descripteur et qui est basé sur le principe de la sélection par cohérence. Avec chaque correspondance, nous avons la rotation éventuelle et le changement d'échelle entre les descripteurs. L'idée générale est d'identifier les correspondances correctes en étudiant la rotation et le changement d'échelle entre deux images. L'angle de rotation est extrait par différence entre les angles des deux points d'intérêts qui font l'objet d'une correspondance. Puis le facteur d'échelle est calculé de deux manières différentes : soit on utilise le facteur idéal α défini à l'équation 4.4, soit on calcule directement le ratio des distances entre les points retenus dans les secteurs et le point central, les distances d_i.



(a)



(b)

FIGURE 4.2 – Accumulateur. (a) Distribution obtenue en utilisant α (voir équation 4.4), et dans la bulle, distribution obtenue en utilisant le ratio des distances. Nous avons une rotation de 30 degrés et un facteur d'échelle de 1,0 pour plus de 450 bons appariements. (b) Distribution obtenue lorsqu'aucune correspondance correcte est possible.

Ensuite, l'angle de rotation et le facteur d'échelle sont directement injectés dans l'accumulateur correspondant à une matrice où chaque cellule représente un angle de rotation et un facteur d'échelle possible. Plusieurs correspondances peuvent être prises en compte pour un même point d'intérêt. Les mauvaises correspondances se répartissent un peu partout dans la matrice et les bonnes se concentreront autour d'une rotation et d'un facteur d'échelle comme illustré dans la figure 4.2 (a). Dans le cas où deux images ne peuvent pas être appariées, aucun pic ne se dégage de l'accumulateur (voir la figure 4.2 (b)). La détermination de la bonne correspondance se fait donc par accumulation d'évidences.

Une discrétisation des angles de rotation et des facteurs d'échelle est nécessaire. La nature du descripteur ne nous permet pas d'atteindre une invariance en changement d'échelle importante.

Ainsi, les facteurs d'échelle possibles se situent entre 1,3 et 0,75. Nous pouvons également, selon l'application, adapter l'intervalle des angles possibles.

Enfin, notre accumulateur permet d'une part de retenir uniquement les bonnes correspondances mais aussi d'avoir une estimation de la rotation et du facteur d'échelle entre deux images. La condition préalable est que les images ne doivent pas avoir de changements de points de vue (rotation de la caméra sur les plans non parallèles à la scène). De plus, une seule correspondance correcte entre des descripteurs possédant au moins 4 points permet de retrouver complètement la transformation (homographie) entre 2 images et 3 points suffisent pour retrouver une transformation affine. Cela nous permet d'être assez strict sur les conditions d'appariement entre descripteurs.

4.4 Évaluation

4.4.1 Base de données

Bien qu'il existe plusieurs bases de données pour évaluer la qualité d'un algorithme complet d'appariement, nous avons choisi d'évaluer notre algorithme sur notre propre

base de données. En effet, ce choix est justifié par la spécificité de notre méthode. Nous voulons comparer surtout les algorithmes existants sur leurs capacités à apparier correctement 2 images avec très peu de textures et/ou très peu de contrastes. Ainsi, nous avons alimenté notre base de données sur ce type d'images. Il s'agit d'images de surface en béton que l'on retrouvera aussi par la suite dans l'application de reconnaissance de défauts sur les ouvrages d'art. Les images choisies sont de différentes tailles et ont été prises avec différentes rotations et à différentes échelles. Au total 20 paires d'images sont utilisées dans l'évaluation. Nous pouvons observer sur la figure 4.4 des exemples de la base de données.

4.4.2 Critère d'évaluation

La plupart des images sont peu texturées. Cependant sur les images présentant plus d'éléments, plus de correspondances ont été trouvées. Nous avons également comparé notre algorithme par accumulateur à d'autres algorithmes limitant la confusion en utilisant avec notre descripteur la méthode RANSAC.

Pour le paramétrage spécifiques aux autres méthodes, la région R choisie pour le descripteur FLASH va de 20 pixels à 100 pixels. Pour SIFT, les paramètres par défaut (ceux présentés dans [Lowe04]) ont été utilisés. Pour les méthodes de ORB et BRISK utilisant une méthode inspirée de FAST, nous avons fixé le seuil à 10.

Les métriques d'évaluation des descripteurs sont les mêmes que celles utilisées dans la littérature notamment dans [Mikolajczyk05]. La *precision* est définie par

$$precision = \frac{\#correspondances\ correctes}{\#putatives\ correspondances} \tag{4.9}$$

et le *recall*

$$recall = \frac{\#correspondances\ correctes}{\#possibles\ correspondances} \tag{4.10}$$

Les détecteurs de points d'intérêt sont évalués avec la *répétabilité*. Elle est définie par le pourcentage de points qui a été trouvé dans 2 images représentant la même scène mais avec des points de vues différents. Ainsi ces points représentent les mêmes points

physiques dans la réalité. Son expression est la suivante :

$$repetabilite = \frac{nombre\ de\ pairs\ en\ commun}{min(points\ dans\ les\ 2\ images)} \tag{4.11}$$

Ce calcul est réalisé uniquement sur les parties des images présentes dans les 2 images.

4.4.3 Taux de reconnaissance

Dans la figure 4.3, nous constatons que le descripteur FLASH a une précision importante car il contient des informations sélectionnées soigneusement qui donnent une bonne idée de la structure du voisinage. En conséquence, le rappel est très bas comparé aux autres algorithmes. Cependant, les informations contenues dans les correspondances de FLASH sont plus robustes et permettent d'estimer la transformation géométrique entre 2 images.

Afin de minimiser le temps d'exécution, nous avons éliminé très tôt les points peu distinctifs en comparant uniquement les chaines binaires. De plus, les points qui n'apparaissent pas dans les 2 images simultanément sont la cause de confusion pour la chaine binaire du descripteur FLASH. Notre accumulateur possède une meilleure précision que RANSAC qui est dans ce cas proche de la version dégradée de FLASH. Notre accumulateur permet donc une bonne estimation du type de transformation qu'il y a entre 2 images. Le point négatif est que la robustesse de notre algorithme est limitée à certaines transformations comme le changement d'échelle ou la rotation. L'élaboration de notre descripteur n'est pas destinée à être robuste aux changements de perspectives (ou points de vue). Par exemple, on ne gère pas le cas où un cercle serait vu de travers comme une ellipse. Il s'agit là d'une extension possible à notre descripteur mais que nous n'avons pas abordée dans le cadre de cette thèse. Nous nous rattachons à notre problématique de détection de défauts sur ouvrages d'art où les photos sont prises successivement et parallèlement au plan d'un mur plan.

Bien que la répétabilité soit insuffisante, SIFT a un bon rappel et moins de 50% de précision. Cela explique pourquoi l'utilisation de la méthode RANSAC est rendue dif-

ficile. ORB et BRISK, qui ont des répétabilités similaires par leur inspiration commune, ont une précision moins importante que FLASH. L'association des correspondances nous donne facilement et presque immédiatement une estimation de la transformation entre 2 images.

La version dégradée du descripteur FLASH peut être utilisée sous des contraintes de temps fortes. Les résultats montrent une diminution dans la précision et dans le rappel, cependant la précision reste meilleure que ORB qui reste très répandue dans les applications d'appariement en temps réel.

Dans les 2 versions du descripteur FLASH, nous pouvons évaluer la transformation entre 2 images. Dans l'appariement d'images, il est important de rappeler que le but est de trouver la transformation entre 2 images et non d'avoir le plus de bonnes correspondances. Le nombre de points traités n'est donc pas primordial comme le montre l'image au centre de la figure 4.4 où une seule bonne correspondance a été trouvée. Avec peu de points, comme nous l'avons déjà dit, un appariement peut être fait en construisant un seul descripteur FLASH.

4.4.4 Temps d'exécution

Le temps d'exécution a été enregistré avec un ordinateur doté d'un processeur Intel i7-6700 2.60 Ghz sur un Windows 10 64 bits. L'implémentation des algorithmes de la littérature est fournie par la librairie OpenCV en utilisant les optimisations SIMD(Single-Instruction-Multiple-Data) comme SSE2 ou AVX. Ces optimisations permettent de réduire le temps d'exécution des algorithmes. Nous n'avons pas pu refaire les implémentations de chaque méthode mais nous avons continué à comparer les temps d'exécution malgré cela. Il est donc important de souligner que notre implémentation ne bénéficie d'aucune optimisation. D'ailleurs, un seul coeur du processeur est utilisé pour l'implémentation de toutes les composantes de notre méthode (détection, description, méthodes d'appariement).

Notons que nous pouvons envisager un traitement en parallèle pour améliorer notre méthode. En terme d'ingénierie, il faudra surtout penser à l'accès aux ressources

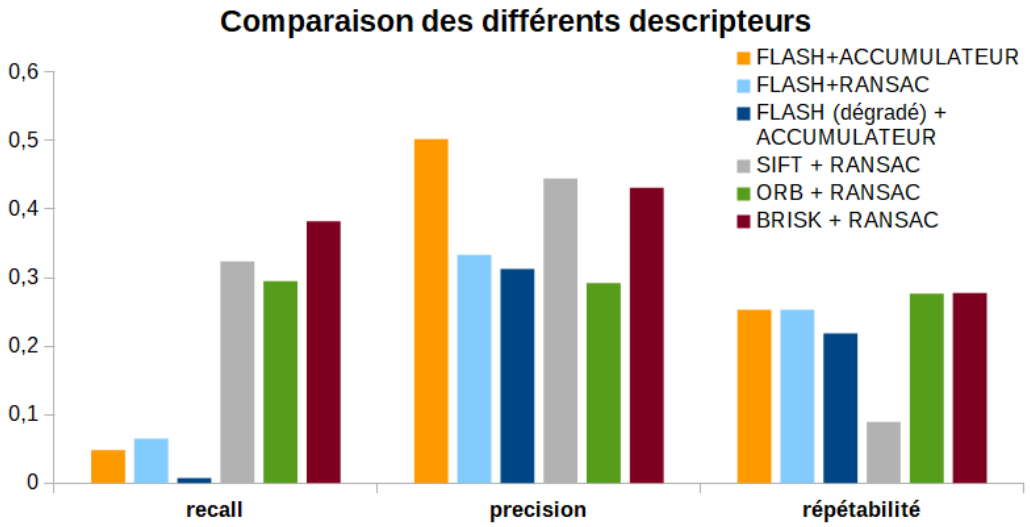


FIGURE 4.3 – Comparaison des différents descripteurs. 20 paires d'images sont utilisées pour l'évaluation.

Processus complet	(CPU optimisations)	ORB	BRISK	**FLASH**
Temps ($\times 10^7$ s/pixel)	3.86	0.12	1.43	1.39 **(0.5)**

TABLE 4.1 – Résultats en temps d'exécution (en gras le temps d'exécution avec le score dégradé)

concurrentes durant la construction des descripteurs et ainsi créer d'abord un découpage intelligent de l'image.

Afin d'améliorer l'exactitude du descripteur FLASH, nous utilisons des calculs de distances et de ratios. Avec la description complète FLASH, le temps d'exécution est proche d'un SIFT **optimisé** (voir tableau 4.1). Ces calculs ne sont pas favorables au temps de traitement. Avec la version dégradée nous passons sous 1 seconde pour traiter une image. ORB est plus efficace en temps et c'est aussi la raison pour laquelle il est très utilisé aujourd'hui. Cependant, FLASH peut encore faire des progrès en exploitant le parallélisme.

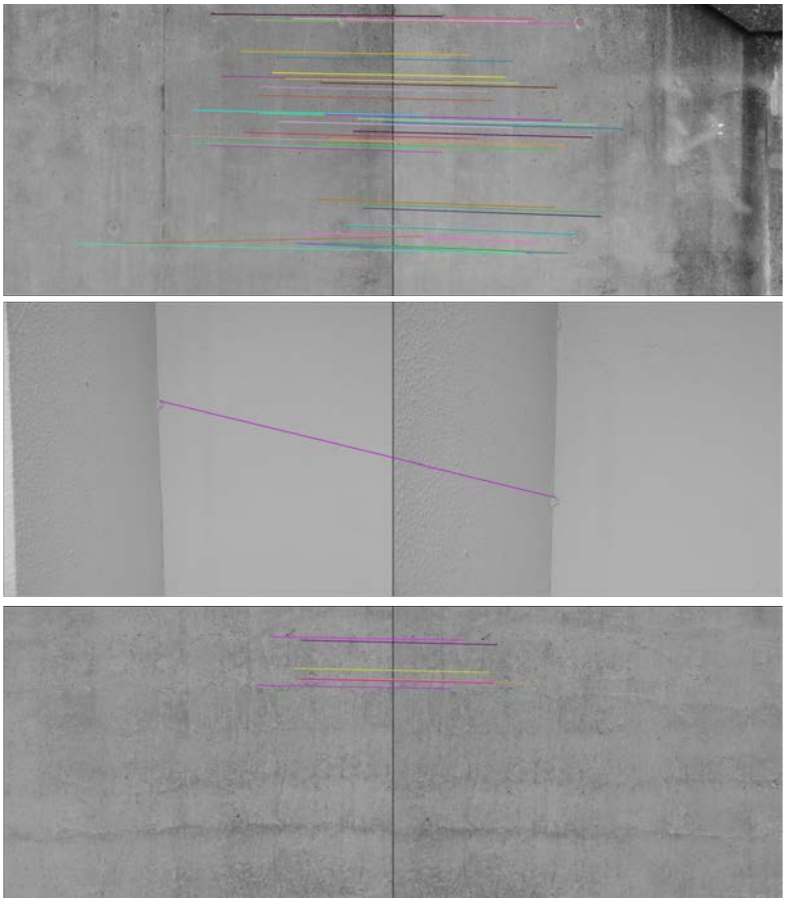


FIGURE 4.4 – Exemples d'appariement sur les images de notre base de données pour l'évaluation. Plus de 90% des correspondances illustrées sont correctes.

4.5 Conclusion

L'appariement d'images sert dans différents domaines comme le SLAM, la reconnaissance de documents, le suivi d'objets. Dans notre projet de détection de défauts sur ouvrages d'art, le besoin s'exprime par la capacité à reconnaître un défaut apparaissant sur plusieurs images. Cela permettra d'éviter d'identifier un défaut plusieurs fois et permettra d'aider à une meilleure détection en fusionnant les informations. Par exemple, on peut suivre une longue fissure qui aurait son tracé sur plusieurs images.

Dans ce chapitre, nous avons présenté différents algorithmes existants destinés à l'appariement d'images. Ensuite, nous avons proposé un détecteur inspiré de FAST en ajoutant des points FLASH plus stables. En l'associant avec notre descripteur, nous avons un algorithme d'appariement qui permet d'obtenir directement les bonnes correspondances. De plus, une seule correspondance suffit à établir la transformation entre deux images. FLASH obtient une bonne précision dans des images peu texturées et est invariant en rotation, aux occlusions, et à une plage de changements d'échelle. Notre descripteur n'est pas robuste aux changements de point de vue si ce changement est trop important. Beaucoup d'améliorations sont possibles notamment sur les temps de calculs. Cependant, les versions proposées permettent déjà d'avoir un algorithme qui peut être embarqué sur un vecteur d'acquisition. Ces algorithmes ont permis une application dans l'analyse de documents que nous présentons dans un chapitre suivant.

5

Détection de défauts : les défauts surfaciques

5.1 Introduction

Les défauts (ou avaries) surfaciques représentent une catégorie de défauts liés à une dégradation mécanique (chocs, tension) ou chimiques (alcali réaction, gonflement). Tous ces défauts présentent des caractéristiques très différentes en texture, en forme et en couleur. Cela peut dépendre de plusieurs facteurs difficiles à identifier dans notre cas. Les types de défauts de surface les plus répandus sont les éclatements avec/sans fer apparent, l'humidité, la végétation. Le graffiti peut être également considéré comme défaut car il s'agit d'une dégradation visuelle de la surface. Elle peut avoir pour conséquence de réduire le contraste d'une fissure lorsque celle-ci était déjà présente au moment de l'application de la peinture aérosol. La surveillance d'infrastructures implique une détection de tous les types de défauts portant atteinte à la santé d'un ouvrage. Nous avons deux stratégies possibles : la détection à partir des caractéristiques de défauts connus ou la détection de l'anormalité à partir de surfaces sans défaut.

La détection d'anomalies inclut l'anormalité, la nouveauté ou la détection d'aberrations. Dans chaque cas, le but est de détecter une situation différente de la normale. Comme domaines d'applications possibles, on peut mentionner la détection de comportement anormal en vidéo-surveillance ou la détection d'anomalies dans les images. Dans ce contexte, les instances représentant la "normalité" sont disponibles. Les autres instances, appartenant à une classe négative, ne sont pas présentes ou pas bien définies dans le jeu de données.
Ainsi, la classification one-class est une problématique importante. Modéliser la normalité est une partie essentielle dans le processus de détection. Distinguer les situations dites "normales" des autres est donc un verrou scientifique important qui peut être envisagé sous l'angle d'une problématique de classification indépendante one-class.

L'utilisation d'une seule classe peut être imposée dès lors qu'on ne dispose pas de suffisamment d'informations sur les autres situations. Dans notre domaine, les défauts représentent une très faible partie d'une infrastructure entière. La détection des défauts est utile et mène vers d'importantes décisions de maintenance. La difficulté dans la

surveillance d'ouvrages d'art est le nombre important de défauts existants. De plus, un même type de défaut peut être visualisé de manière très différente.

Nous proposons ici un système reposant sur une analyse par zones, utilisant des descripteurs et des méthodes d'apprentissage. Nous proposons une solution capable d'améliorer la capacité d'un réseau de neurones convolutif à détecter des anomalies, en injectant des descripteurs statistiques comme les LBP. Cette idée est testée dans le cadre spécifique de la détection des éclatements de béton.

Bien que des descripteurs peuvent être sélectionnés de façon automatique, nous avons d'abord étudié les contraintes inhérentes au projet : la nature de la surface rencontrée (la surface a-t-elle une texture orientée, régulière ou irrégulière ? la couleur est-elle un élément essentiel ?), les conditions d'acquisitions (illumination, perpendicularité à la surface,...) les contraintes d'application (la disponibilité des ressources en mémoire, les temps de calcul, la technologie utilisée, cherche-t-on une localisation des défauts ...) L'étude des descripteurs à la classification binaire a montré que nous pouvions a priori, réduire l'espace des descripteurs testés.

Nous allons présenter différentes méthodes de caractérisation de texture pouvant être utilisées pour réaliser la détection des défauts en se basant sur la texture et la couleur.

La partie d'état de l'art du chapitre est divisée en deux grandes parties, portant d'abord sur une généralisation pour tous types de défauts des systèmes de détection et de classification, puis une spécialisation de la détection pour les éclatements de béton (ou épaufrures). Le chapitre présente ensuite une proposition d'architecture complète pour la détection de défauts surfaciques dédiée aux éclatements de béton.

5.2 État de l'art sur la caractérisation de texture et la détection de défauts

5.2.1 Caractérisation de texture

Les techniques présentées dans la section suivante ne sont pas utilisées directement pour la détection de défauts. Comme nous l'avons souligné, les défauts sont essentiellement caractérisés par des différences de texture, c'est la raison pour laquelle nous nous focalisons dans cet état de l'art sur les travaux portant sur l'analyse de texture pour la segmentation d'images appliquée à divers domaines d'applications de la vision par ordinateur.

5.2.1.1 Méthodes structurelles ou géométriques

Dans cette famille de méthodes, les éléments structurant la texture sont étudiés en terme de disposition et de répétition dans une zone fixée. Pour cela, ces méthodes se basent sur la détection de lignes, de coins, de contours etc. que l'on appelle des primitives. Ensuite, les propriétés locales des primitives et les relations entre ces dernières sont prises en compte comme descripteurs. Par exemple, on peut étudier la forme, la taille ou l'orientation d'une primitive ainsi que son homogénéité ou si elle est placée à coté d'une autre primitive.
Les méthodes structurelles sont surtout dédiées à l'analyse des textures très régulières (comme les textures synthétiques). Par conséquent, elles ne sont pas adaptées dans notre application car une surface de béton présente une texture irrégulière. Des exemples d'applications sont décrits dans [Humeau-Heurtier19].

5.2.1.2 Méthodes spatio-fréquentielles

Les méthodes spatio-fréquentielles cherchent à analyser les fréquences d'une zone localisée d'une image. La transformée de Fourier est utilisée uniquement dans

le domaine fréquentiel afin, par exemple d'identifier des empreintes digitales (voir [Humeau-Heurtier19]). Généralement, les domaines fréquentiel et spatial sont combinés donnant ainsi des descripteurs connus comme les filtres de Gabor [Daugman85] ou les ondelettes et ses variantes. De plus, les calculs dans le domaine fréquentiel présentent un intérêt certain pour la vitesse d'exécution. Très tôt, à la fin des années 80, les filtres sont utilisés comme caractéristiques qui discriminent les différentes textures, notamment dans le domaine médical. Comme les bancs de filtres de Gabor, les ondelettes découvertes par Jean Morlet, permettent d'analyser une image à plusieurs échelles, en réalisant de manière répéter les opérations de filtrages. Ces filtrages réalisés permettent d'extraire des caractéristiques telles que l'énergie, l'entropie, la variance ou des histogrammes. D'autres alternatives aux ondelettes sont décrites dans [Humeau-Heurtier19] comme les contourlets ou shearlets.

Bien que les outils d'analyse de textures basés sur l'étude du spectre de l'image et ses différentes décompositions, soient très utilisés par la communauté, dans notre cas, il présente des inconvénients relatifs à l'invariance en rotation ou la détermination du bon facteur d'échelle.

5.2.1.3 Méthodes basées sur l'apprentissage

L'apprentissage de vocabulaire visuel et les mécanismes basés sur les réseaux de neurones font partis des méthodes plus récentes de caractérisation de texture. Les réseaux de neurones peuvent être à convolution et sont plus ou moins complexes allant de l'extreme learning machine aux réseaux à plusieurs couches comme AlexNet ou VGG-16 (voir [Andrearczyk17] pour plus de détails). Ces approches d'analyse de texture sont très exploitées dans le domaine médical [Cimpoi14] ou pour la description de texture dans divers contextes industriels dans [Andrearczyk17]. Ces méthodes récentes sont très performantes mais elles nécessitent des temps de calculs importants et elles sont dépendantes de leurs bases d'apprentissage. Cependant une étude proposée par Liu en 2016 dans [Liu16a], montre que les LBP et ses variantes restent plus performantes que les réseaux de neurones profonds sur l'ensemble des jeux de données

testées et spécifiquement pour des textures présentant de faibles variations.

5.2.1.4 Méthodes statistiques

Les méthodes statistiques peuvent se catégoriser selon les descripteurs statistiques du premier ordre, du second ordre ou d'ordre supérieur qu'elles exploitent. La moyenne, la variance, ou l'entropie, et les histogrammes sont des mesures du premier ordre calculées directement sur l'image. Les approches du second ordre sont plus complexes et font parties des premières approches utilisées pour discriminer des textures. On rappelera les ensembles de descripteurs obtenus à partir de matrices de cooccurrences ou GLCM (Gray Level Co-occurrence Matrix) de Haralick [Haralick 73]. Ces descripteurs ont pour objectif de déterminer des caractéristiques d'arrangement et de propriétés spatiales d'une image. A partir de ces matrices, on calcule différentes mesures statistiques comme le moment du second ordre, le contraste, la corrélation, la variance ou encore l'entropie. Ce type d'approche très coûteux en temps de calcul et il n'est pas adapté au système embarqué tel que nous l'envisageons. On notera également que les méthodes statistiques du premier ordre n'ont pas la robustesse nécessaire au niveau du changement d'illumination ou de la rotation tel que cela est requis sur des images de texture de béton.

Dans cette famille de méthodes, on peut également mentionner les techniques basées sur l'extraction de motifs binaires locaux (LBP) très exploitées dans la caractérisation de texture. Dans [Ojala02], les LBP permettent de classifier plusieurs textures comme l'herbe, le rotin, le cuir, le bois ou la laine. Dans [Bensoltana14], les LBP permettent de catégoriser des images de dentelles en utilisant l'histogramme des motifs présents dans l'image. Ils sont utilisés également pour la reconnaissance faciale dans [Levi15] et [Liu16b]. La description de ces motifs est présente dans la section 3.2.2.

5.2.1.5 Autres méthodes basées graphe

Les méthodes basées sur les graphes sont souvent invariantes à la rotation et à l'illumination. Selon [Humeau-Heurtier19], les résultats montrent que ces méthodes sont robustes au bruit. Cependant, elles ne permettent pas, dans la plupart du temps, une exploitation facile des descripteurs avec d'autres systèmes et/ou d'autres descripteurs.

Dans nos propositions, nous avons choisi d'exploiter les descripteurs liés aux LBP. Les contraintes du projet nous permettent de ne pas se soucier du problème de la sensibilité au changement d'échelle. Son invariance à l'illumination est un atout lorsque des images d'une même scène sont acquises à des heures d'ensoleillement différentes. De plus, les LBPs ont montré, selon [Humeau-Heurtier19], qu'il était possible d'exploiter ces descripteurs de différentes manières. De nombreux auteurs se sont intéressés à déterminer des variantes permettant de les combiner à d'autres descripteurs [Bensoltana14]. Aussi, les premières contributions sur la détection de fissures nous ont conduit à l'utilisation des descripteurs LBP à partir du descripteur local FLASH que nous avons proposé.

5.2.2 Analyse de défauts sur les surfaces de béton

FIGURE 5.1 – (Gauche) Défaut avec fer apparent (Droite) Défaut sans fer apparent.

Les types de défauts de surface les plus répandus sont les éclatements avec ou sans fer apparent (voir figure 5.1), l'humidité, la végétation. Le graffiti peut être également considéré comme un défaut car il s'agit d'une dégradation visuelle de la surface. Elle peut avoir pour conséquence de réduire le contraste d'une fissure lorsque celle-ci était

déjà présente au moment de l'application de la peinture aérosol. Certains de ces défauts ne sont pas traités dans la littérature et différentes techniques sont utilisées pour chacun d'eux. Nous présentons ici des algorithmes de détection de tous types de défauts surfaciques. Les fissures sont parfois détectées au même titre que les défauts de surface. Mais ici, nous ne considérons pas les fissures qui ont fait l'objet du chapitre 2.

5.2.2.1 Approches bas-niveau

La détection de défauts est une thématique de recherche de la vision par ordinateur qui a suscité beaucoup d'engouement dans l'industrie notamment pour le contrôle qualité des matériaux. Elle permet de poser un diagnostic de présence d'un défaut sur une image sans nécessairement préciser la localisation. Dans [Zhu08] par exemple, les auteurs ont exploité des techniques classiques de traitement d'images comme les filtres gaussiens pour la détection de défauts surfaciques de type de bulles d'air dans les surfaces de béton. Les écarts-types de niveaux de gris sont ensuite utilisés pour détecter des traces d'humidité. On dénombre ainsi de nombreux travaux relevant d'approches variationnelles basées sur des indices locaux de luminance ou de couleurs, [Koch11]. Dans [German12], la détection d'éclatements de béton est exécutée à l'aide du seuillage par entropie locale. L'estimation de la surface qu'occupe le défaut ainsi que la détection de fer apparent par analyse colorimétrique permet de compléter l'analyse.

5.2.2.2 Approches par réseaux convolutifs simples

Aujourd'hui, les approches de détection de défauts reposant sur des méthodologies d'apprentissage automatique sont davantage utilisées pour faciliter le paramétrage des systèmes et augmenter leur adaptativité. Un comparatif de méthodes de détection de défauts sur des surfaces de béton a été proposé en 2012 dans [Son Hyojoo12] à partir de la prise en compte d'espaces colorimétriques sur une série d'expérimentations reposant sur trois algorithmes de classification (mixture de gaussiennes, réseaux de neurones artificiels et SVM). C'est à partir de 2015, que les approches convolutionnelles ont été privilégiées dans ce contexte. On mentionnera, par exemple les travaux de Makanta-

sis dans [Makantasis15] qui a exploité un réseau convolutionnel simple composé de 2 couches de convolution et 3 couches complètement connectées de perceptron. En entrée du réseau, des caractéristiques variationnelles bas-niveau sont calculées. Chaque pixel est représenté par un vecteur où l'on retrouve l'entropie, des coefficients issus d'un banc de filtres de Gabor pour la texture, la fréquence obtenue par application du Laplacien et également des informations concernant les contours. Le réseau produit une décision binaire de la présence d'un défaut pour chaque pixel.

Le *Fully Convolutional Network* inspiré de [Shelhamer16] est employé afin de segmenter les défauts dans [Xiao18]. Le réseau est composé uniquement de couches de convolution et réalise une segmentation de l'image. Cette dernière permet en même temps d'obtenir une classification au niveau pixel des pièces d'estampage galvanisées défectueuses. Aussi, une segmentation au niveau pixel est proposée dans [Tao18] pour uniquement la détection sur des pièves métalliques. Le réseau de neurones est un auto-encodeur, c'est à dire un modèle génératif qui a été entrainé pour retourner une image où les pixels défauts sont différenciés de ceux sans défaut. Un autre réseau plus compact est proposé afin de classer les zones considérées avec défauts. Trois sorties sont possibles pour ce réseau : éraflure, poussière et zone endommagée.

Dans l'étude menée dans [Koch15], les auteurs montrent que les fissures et les éclatements sont des défauts traités largement dans la littérature. Cependant les autres types de défauts ne font pas l'objet de travaux majeurs et sont alors abordés par très peu d'auteurs(voir pas du tout). Par exemple, aussi loin que la recherche bibliographique nous a menés, nous n'avons effectivement pas trouvé de solutions proposées pour les défauts liés à la végétation.

5.2.2.3 *Region proposal networks*

Plusieurs méthodes de requalification de CNN sont également proposées pour détecter et localiser des objets par propositions de régions. Différentes implémentations et améliorations ont été proposées dans la littérature([Girshick15], [Ren16], [Redmon16]). Dans [Cha17b], ils accomplissent une détection très spécifique comme les écrous corro-

dés et la délamination de fer. Dans le but de réduire le nombre d'exemples durant la phase d'apprentissage, les techniques de *transfer learning* avec le réseau VGG-16 sont exploitées pour la détection d'éclatements dans [Gao18]. Plus récemment, [Xue18] utilise les CNNs afin de détecter et de classifier trois sortes de défauts : les fissures, les fers apparents et les cavités. L'architecture du réseau est inspirée de Faster R-CNN [Ren16]. Quelques modifications d'architecture sont proposées afin d'améliorer la classification.

5.2.2.4 Les jeux de données d'entraînement

Peu de bases de données annotées portant sur la détection de défauts (en open data) sont rendues disponibles à la communauté scientifique. Les images sont le plus souvent privées comme le soulignent les auteurs dans [Koch15]. Une base annotée disponible est la CSSC (Concrete Structure Spalling and Crack database [1]) Cependant, elle n'est pas représentative de la réalité et de la diversité des défauts du fait de la piètre qualité des photographies. Elle contient près de 278 images d'éclatements de béton labellisées à la main à l'aide d'experts et a été construite par une fouille d'images sur internet. Ces images ont été trouvées en inscrivant plusieurs mots-clés. Par exemple, on a *Concrete spalling, Concrete delamination, concrete bridge spalling, concrete column spalling, concrete spalling from fire*. La partie fissure de cette base a été construite de la même manière et contient 954 images. Il s'agit pour beaucoup de photos avec des résolutions et dimensions totalement différentes. De plus, beaucoup d'autres éléments apparaissent tels que des machines de travaux ou des personnes. Les auteurs se sont focalisés sur les fissures et les éclatements de béton.

Beaucoup de techniques dans la littérature sont spécialisées sur un type de défaut. Il est nécessaire de disposer de données annotées en quantité suffisante mais le constat récurrent est que celles-ci sont difficilement accessibles publiquement. C'est la raison pour laquelle, nous avons choisi de développer une approche de la détection de défauts utilisant exclusivement les exemples de défauts surfaciques et de surfaces sans défaut en ciblant ainsi des catégories très spécifiques. Notons enfin que dans le cas des défauts

1. https ://github.com/ccny-ros-pkg/concreteIn_inpection_VGGF

surfaciques, les classes de défauts ne sont généralement pas connues a priori. Il est nécessaire de se pencher vers la conception d'un système de reconnaissance dynamique ayant la capacité à s'auto-adapter aux nouvelles données.

L'état de l'art sur la détection de défaut montre qu'il existe peu de solutions qui tiennent compte à la fois de l'évolution de défauts selon la structure qui les contient, du nombre d'échantillons disponibles, et de la qualité des sources. Par ailleurs, les outils de diagnostic et de classification développés reposent pour la plupart sur les interactions existantes entre les images contenant des défauts de celles qui n'en ont pas (cas des classificateurs discriminants). Du fait de l'entremêlement des informations exploitées dans les modèles discriminants, l'évolution des classes de défauts (dont la fusion ou la division de classes), et la détection automatique de nouvelles classes deviennent plus délicates. A contrario, alors que les classificateurs dits indépendants (commes les SVM one-class) semblent plus naturellement adaptés à ces types de problèmes, leur usage reste très modeste, sans doute à cause de la difficulté de leur paramétrisation et de leurs performances souvent moins élevées en classification que leurs homologues discriminants.

Ces observations nous ont conduits à privilégier une classification indépendante (One-Class SVM), permettant de traiter des données de catégories non équilibrées (*unbalanced data*), des données au départ inexistantes (classes inconnues rajoutées à la connaissance du système au fur et à mesure), des données évolutives (classes de défauts pouvant évoluer selon les acquisitions faites au fur et à mesure et acceptant une plus grande diversité de supports, de lieux, et de qualité d'images).

Cette approche présente également un intérêt important pour le ré-entraînement du système qui peut s'effectuer par parties seulement et de manière incrémentale : il n'est pas nécessaire de relancer un apprentissage complet pour le faire évoluer et améliorer la reconnaissance.

5.2.3 Apport de la classification one-class pour la détection d'anomalies

Les machines à vecteurs de support (SVM)[Vapnik98] sont des algorithmes d'apprentissage supervisé utilisés pour la classification ou pour des tâches de régression. Un modèle SVM est entraîné afin de trouver une séparation linéaire optimale entre les classes. Cette séparation correspond à un hyperplan dans l'espace des descripteurs et doit maximiser la distance entre les classes.

Les one-class SVM (OC-SVM) ont été popularisés par [Yunqiang Chen01] mais apparaissent la première fois dans [Schölkopf01]. Pour les one-class SVM, une classe positive définit les données d'apprentissage et aucune donnée de classe négative n'est exploitée. Dans l'espace de représentation, une séparation avec un hyperplan doit être trouvée. On fait l'hypothèse que la distribution de la classe négative dans cette espace est une distribution autour du zéro. On revient alors au problème de départ (classification binaire) avec quelques modifications. Les OC-SVMs apparaissent dans beaucoup d'applications mais récemment, d'autres approches basées sur les réseaux de neurones sont proposées. Faire des hypothèses sur la classe négative est une tâche ardue. Dans [Oza19], la classe négative est assimilée à une distribution normale centrée sur l'origine (comme originellement pour les OC-SVM). Leur réseau est un réseau classique à convolution. L'avant dernière couche simule la classe négative alors que la dernière couche est chargée de faire la différence entre la classe visée (classe positive) et la classe négative créée. Deux sorties donnent la probabilité pour un échantillon d'être dans une classe ou dans l'autre. Cette étape est ajoutée uniquement pendant la phase d'apprentissage.

Dans [Sabokrou18], un réseau antagoniste génératif (GAN) est utilisé pour la détection de nouveauté qui peut être suivie par une classification des échantillons inconnus. On peut assimiler cela à une méthode de classification hiérarchique. Rappelons que dans le cas des défauts surfaciques, les classes de défauts ne sont généralement pas connues a priori. Par conséquent, il est nécessaire de proposer une architecture dynamique ayant la capacité à s'auto-adapter aux nouvelles données.

Le mot *dynamique* se réfère aux changements qui peuvent s'opérer au niveau de la conception des classes ou du classifieur. Les environnements non stationnaires doivent pouvoir tenir compte d'une dérive dans la représentation des différentes classes. Les environnements stationnaires, quant à eux, impliquent une évolution dans le concept de chaque classe, mais la connaissance acquise est conservée dans le temps. Dans chacun des environnements, des nouvelles classes peuvent apparaître et le système doit répondre aux nouvelles contraintes.

Dans le cas de la classification des défauts, nous sommes face à un environnement stationnaire avec la possibilité d'ajouter de nouvelles classes. Les travaux de Khoi dans [Ngo Ho17] ont montré la possibilité de faire évoluer un système de reconnaissance avec les one-class SVMs.

Les auteurs y ont proposé un système dynamique avec plusieurs OC-SVMs. La classification des différents objets est alors indépendante et permet plus de facilité pour faire évoluer les concepts.

5.3 Architecture proposée

L'architecture de reconnaissance de défauts surfaciques est basée sur un framework représenté à la figure 5.2. Notre architecture est similaire à une classification hiérarchique à deux niveaux. En plus d'utiliser les convolutions d'images au sein du GAN initial, elle exploite également les descripteurs LBP "basés FLASH" définis au chapitre 2 ainsi qu'une série de classifieurs one-class SVM évolutifs.

Initialement, c'est un réseau de neurones antagoniste génératif conditionnel (cGAN) qui est utilisé pour distinguer au sein des séries d'images acquises en mobilité celles qui contiennent des défauts de celles qui n'en contiennent pas. Les images prises par drone sont attendues avec une haute résolution (supérieur à 6000x4000 pixels pour 3x2 mètres). Par conséquent, nous avons divisé l'image en patchs d'une taille 64x64 pixels. Le choix de la taille est motivé par le fait que certains défauts ne dépassent pas cette zone dans une image acquise dans les conditions prévues.

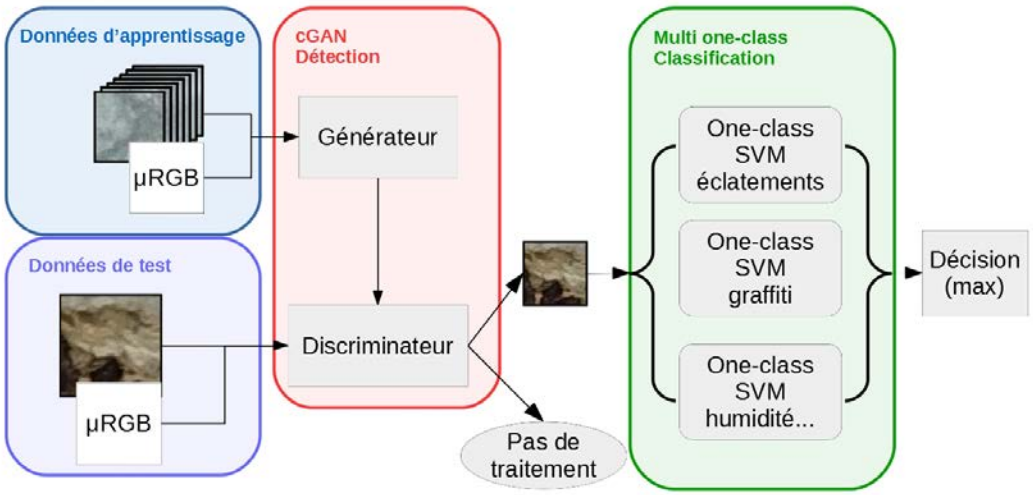


FIGURE 5.2 – Architecture hiérarchique pour la détection et la classification de défauts surfaciques.

5.3.1 Détection de la présence de défauts par cGAN

5.3.1.1 Réseaux antagonistes génératifs conditionnels (cGAN)

Les réseaux antagonistes génératifs sont introduits par [Goodfellow14] et sont une autre façon d'entraîner les modèles génératifs avec la back propagation. Les GANs sont composés de deux réseaux entraînés dans un processus antagoniste similaire à un jeu : un réseau générateur G et un réseau discriminateur D qui évalue la probabilité qu'un exemple a été généré avec G ou est un exemple réel. Comme introduit par [Goodfellow14], G reçoit en entrée un bruit aléatoire. Dans [Mirza14], les auteurs ont proposé de guider la génération afin de créer différentes classes identifiées en introduisant en entrée un vecteur connu et dépendant de sa sortie. Les auteurs réalisent des expérimentations en donnant en entrée les labels de la base de données. Plus tard, la même idée est reprise par [Isola17] dans une implémentation permettant la translation d'images à images.

Dans notre processus, la première tâche est de séparer de manière efficace les patchs

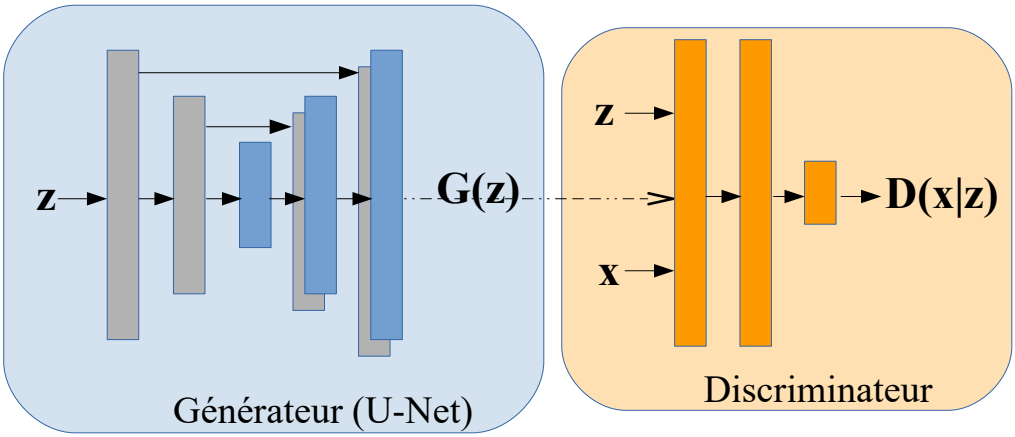


FIGURE 5.3 – Architecture du cGAN.

avec défauts et les patchs sans défaut. Nous avons testé plusieurs méthodes qui sont décrites dans la section 5.4. Sur notre jeu de données, le cGAN est la méthode qui donne les meilleurs résultats de séparation défaut/sans défaut. (voir section 5.4). Notons que les fissures fines ne peuvent pas être détectées avec une structure de type GAN car leurs structures fines sont assimilées à un bruit quelconque dans les images. Le générateur est exclusivement utilisé ici dans le but d'entraîner le discriminateur. Intuitivement, les entrées du générateur doivent avoir suffisamment d'information pour pouvoir reconstruire correctement une image. Le discriminateur pourra alors s'améliorer en créant les bonnes caractéristiques afin d'assurer une bonne classification binaire des images. Cependant, fournir trop d'informations au générateur n'est pas optimal car il sera capable de reconstruire des images complexes dans un processus d'apprentissage très rapide. En conséquence, le discriminateur ne sera pas capable de distinguer les images réelles des images générées car il fera face très vite à des images très similaires aux images réelles. La détection d'anomalies est réalisée uniquement par le discriminateur D.

5.3.1.2 Le détecteur

Nous suivons l'architecture proposée par [Isola17]. Les patchs étant plus petits, nous avons adapté les premières couches de convolution. Dans [Isola17], les auteurs ont utilisé un patchGan d'une taille de 70x70 pixels qui permet au discriminateur de

sortir une matrice de scalaires et non une valeur unique. La décision du discriminateur se fait donc sur un patch et non sur l'image complète. Cela permet d'obtenir des images avec moins de flou pour la génération des images. Ici, notre image est découpée en patchs mais le discriminateur est entraîné comme un CNN classique et il produit une valeur unique pour chaque patch. Ainsi, pour traiter une image entière, cette dernière est divisée en patchs et une décision de détection est prise par chaque patch constituant l'image. Au vue de la difficulté d'obtenir des images labellisées, les résultats sont présentés sur l'ensemble des patchs et non sur les images entières. Le découpage de l'image peut être fait de différentes façons : avec un pourcentage fixé de recouvrement entre les patchs, sans recouvrement ou encore, sur des zones spécifiques de l'image. Plus le recouvrement est important, plus le nombre de patchs à analyser est grand et le temps de calcul est élevé. Ici, nous avons choisi de découper sans recouvrement afin de ne pas augmenter le temps de calcul. Nous n'avons pas étudié l'incidence du découpage sur les résultats, cela reste une perspective à envisager dans la poursuite des travaux. En utilisant uniquement le discriminateur D, les données d'origine x et l'information conditionnelle z montrées sur la figure 5.3, la détection est simple et est définie comme suit :

$$S(x|z) = \begin{cases} 1\ (classe\,vise) & \text{si } D(x|z) > T, \\ 0\ (anormal) & \text{sinon} \end{cases} \tag{5.1}$$

où T est un seuil prédéfini.

L'information conditionnelle z est ici un ensemble de caractéristiques liées à la texture et/ou la couleur. Ces dernières sont décrites dans la partie suivante.

5.3.1.3 Entrées conditionnelles z

Les différents défauts diffèrent par leur couleur et/ou leur texture. Les motifs binaires locaux (LBP) donnent une information de texture de l'image et sont décrits dans la section 3.2.2. En considérant uniquement les motifs considérés comme uniformes, nous disposons de 9 familles de motifs invariants en rotation. Sur la figure 5.4, nous pouvons visualiser la différence statistique entre une image sans défaut et avec défaut.

L'analyse des familles montre que les motifs uniformes sont plus nombreux pour les images sans défaut.

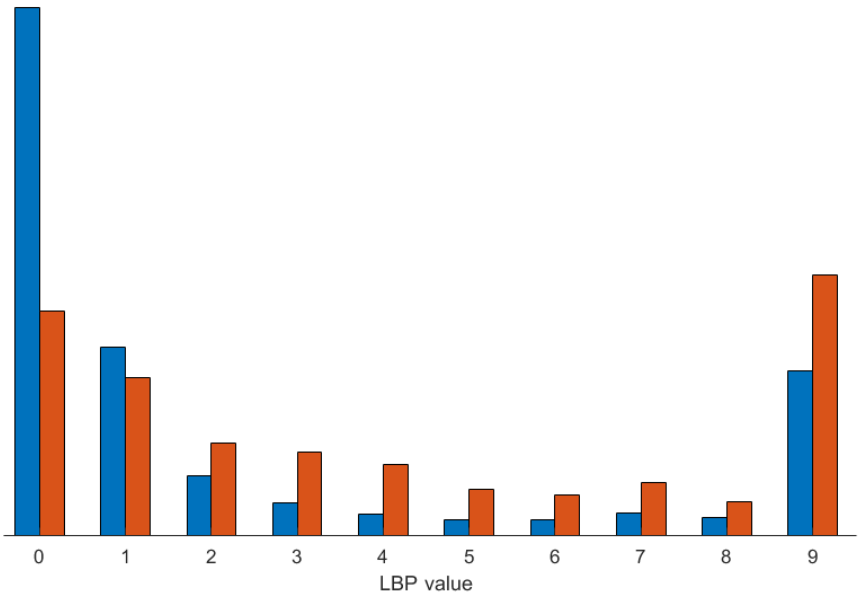


FIGURE 5.4 – Profil de l'opérateur LBP pour une image sans défaut (bleu) et avec défaut (rouge).

Les entrées de notre cGan sont au nombre de N couches : 3 pour une information liée à la colorimétrie et $N-3$ pour la texture créée par les points homogènes des autres machines. Une couche notée, C^i_{lbp}, est définie comme suit :

$$C^i_{lbp} = \begin{cases} 1 & \text{si} \quad LBP^{riu2}_{P,R} = i, \\ 0 & \text{sinon} \end{cases} \tag{5.2}$$

Toutes les entrées possibles, noté E, sont définies comme suit :
$E = \left\{ \mu R, \mu G, \mu B, C^0_{lbp}, \cdots, C^9_{lbp} \right\}$
ou μX est une couche pour laquelle toutes les valeurs sont normalisées à la moyenne du canal correspondant X. En pratique, pour un GAN classique z est un vecteur aléatoire. Ici, nous utilisons un GAN conditionnel (cGAN) avec $z = E$ une entrée connue sans

information aléatoire intégrée. Dans notre cas, z est représenté par des matrices remplies avec une moyenne RGB entre -1 et 1 contenant une information colorimétrique, et/ou des matrices remplies avec des 0 et des 1 pour signaler la présence de points LBP contenant l'information de texture. Les différentes entrées du GAN sont discutées dans la section 5.4.

5.3.2 Classifieur multi one-class

Avec le manque d'exemples sur les défauts et le large panel des avaries que l'on peut rencontrer, nous avons choisi une architecture dynamique pour la classification. Nous avons choisi l'architecture proposée par [Ngo Ho17]. La figure 5.2 décrit cette architecture adaptée à notre application. Plusieurs OC-SVMs avec les mêmes descripteurs sont utilisés pour classifier les différentes catégories de défauts sur les images repérées par le cGAN. Les OC-SVMs sont indépendants et leur apprentissage peut être réalisé à tout moment dès lors qu'un expert intervient. On peut alors faire évoluer continuellement le système et bénéficier de la capacité incrémentale des SVMs.

5.3.2.1 Calcul des descripteurs pour le multi one-class

Comme pour le fonctionnement des GANs, les descripteurs choisis pour les classifieurs One-Class SVM sont également basés sur une information colorimétrique et de texture. Afin de réduire les temps de calcul moyens, nous avons choisi de réutiliser les valeurs déjà calculées par l'opérateur LBP. Les moyennes des canaux R, V et B sont ensuite incluses comme descripteurs pour la couleur. Au total nous avons exploité 13 descripteurs. Ces vecteurs pourront être enrichis de nouvelles variations autour de la couleur et de la texture pour augmenter la précision des résultats de classification (comme par exemple la variance des canaux de couleur). Au total nous avons exploité 13 descripteurs. Nous rappelons que le choix des descripteurs est motivé par la nature du problème. Nous avons vu que les méthodes statistiques étaient plus adaptées dans notre application. De plus, les calculs se font sur des petites zones de l'image. Cependant, ces patchs ne sont pas assez grands pour avoir des valeurs pertinentes lorsque

l'on doit calculer des moyennes et des variances. Pour un meilleur apprentissage des SVMs, un espace de descripteurs pas trop important est préférable car la séparation entre 2 ensembles sera plus facile. L'augmentation du nombre de dimensions entraine des données très éparses dans l'espace des descripteurs.

5.3.3 Prédictions de la classification

Le modèle proposé par [Ngo Ho17] prédit les classes en sélectionnant la valeur maximale en sortie de classification pour chacun des OC-SVM. Cependant, comme les classifieurs sont indépendants, le seuil de décision peut être différent même si les mêmes descripteurs sont utilisés. C'est pourquoi, nous avons décidé de transformer la valeur prédite f_i par le i^{me} one-class, par sa version normalisée et centrée $\hat{f}_i$. Suivant une distribution normale standard cette valeur peut se définir par l'équation suivante :

$$\hat{f}_i = (\frac{f_i - \mu_i}{\sigma_i^2})^2 \tag{5.3}$$

μ_i et σ_i sont respectivement la moyenne et la déviation standard calculées avec le jeu de données d'apprentissage utilisé pour le i^{me} one-class SVM. Ensuite, nous comparons chaque valeur, la valeur la plus proche de 0 étant celle qui représente le plus la classe à assigner.

Avec cette architecture, il est possible de détecter de nouvelles classes en rejetant tout échantillon ayant une valeur très haute. Un seuil peut être déterminé avec les données d'apprentissage.

5.4 Expérimentations

5.4.1 Scénarios pour la détection des défauts

Nous avons sélectionné trois scénarios lors de la phase de test :

Combinaison des entrées	S1	S2	S3
μRGB	✓	✓	
$\mu RGB + C^0_{lbp} + \cdots + C^9_{lbp}$	✓	✓	✓
$\mu RGB + C^0_{lbp} + C^1_{lbp}$	✓	✓	
$C^0_{lbp} + \cdots + C^9_{lbp}$			✓
googlenet			✓

TABLE 5.1 – Descripteurs utilisés pour la détection. L'entrée de S1 et S2 est z. Pour S3, il s'agit des entrées des OC-SVMs. Rappelons que les entrées sont sous forme d'histogrammes et non par couches pour S3.

1. **S1.** Le discriminateur est uniquement utilisé avec différentes entrées conditionnelles z et l'image à tester x suivant les notations de la figure 5.3. Ce scénario est représenté par les courbes pleines dans la figure 5.5.
2. **S2.** Le discriminateur est utilisé comme dans le premier scénario avec les mêmes entrées conditionnelles z. Au lieu de donner l'image x, nous introduisons, comme dans la stratégie adoptée par [Sabokrou18], la sortie du générateur $G(z)$. L'idée est que la génération de l'image avec défaut échoue et le discriminateur est capable de mieux séparer l'ensemble des images. Ce scénario est représenté par les courbes en tirets dans la figure 5.5.
3. **S3.** Le dernier scénario est une comparaison avec des one-class SVM classiques avec différents descripteurs. Ces derniers sont globalement les mêmes que ceux utilisés dans les deux premiers scénarios. L'implémentation diffère par l'injection de l'information. En effet, on utilise un histogramme en entrée des OC-SVMs pour l'information de texture comme présenté sur la figure 5.4. Nous avons ajouté à cela des descripteurs issus du CNN GoogleNet [Szegedy15]. Ce scénario est représenté par les courbes en pointillés dans la figure 5.5.

Pour chaque scénario, nous décrivons dans le tableau 5.1 les différents descripteurs utilisés.

5.4.2 Base de données et métriques

Notre base de données est composée d'images de surface de béton. Ces dernières sont divisées principalement en 3 classes : normal, éclatement et graffitis. On retrouve des exemples d'éclatements sur la figure 5.1. Nous utilisons également des images considérées comme normales mais présentant des ombres afin de tester la robustesse du système. Au total, plus de 30 grandes images sont utilisées permettant d'extraire plus de 16000 patchs. Nous avons divisé le jeu de données d'images dites normales en utilisant approximativement 70% des patchs pour l'apprentissage et le reste pour la phase de test. De la même manière, chaque classe de défaut est divisée en 2 parties. Le tableau 5.2 montre le nombre de patchs utilisés pour les expériences. La répartition de notre base de données se rapproche des proportions d'images sans défaut et avec défauts que l'on peut obtenir lors d'une réelle acquisition.

TABLE 5.2 – Répartition de notre base de données

Classe	Normal	Défaut	
		Éclatement	Graffiti
Apprentissage	12039	500	500
Total	16271	1393	2751

Les métriques utilisées sont l'aire sous la courbe (AUC) et les courbes ROC présentent le taux de bons négatifs (spécificité) par rapport au taux de bons positifs.

5.4.3 Implémentation

Les deux parties du système ont été testées séparément avec les algorithmes de l'état de l'art. La première partie est implémentée avec le framework *Chainer* et les calculs de LBP sont réalisés à l'aide de la librairie OpenCV. Le conditional GAN est entrainé *from scratch* avec les mêmes paramètres utilisés dans [Isola17]. Les fonctions *loss* utilisées sont égalements les mêmes décrites dans [Isola17]. En reprenant les notations de la

section précédente, la fonction objective s'écrit :

$$G^* = \arg\min_G \max_D \quad \mathbb{E}_{x,z}[\log(D(x|z))] + \mathbb{E}_z[\log(1 - D(G(z)|z))] + \lambda \mathbb{E}_{x,z}[\|x - G(z)\|_1]$$

où le terme avec la distance L1 permet au générateur d'être plus proche de la vérité terrain. Le générateur G essaie de minimiser cette fonction alors que le discriminateur D tente de la maximiser.

Concernant la deuxième partie, les hyperparamètres pour les OC-SVMs sont obtenus par optimisation selon l'algorithme *Grid Search* qui consiste à ré-entrainer les classifieurs avec un ensemble de valeurs des hyperparamètres, puis tester les modèles en calculant le score pour une partie de la base utilisée pour une validation. Les tests à destination de l'optimisation peuvent être faits en utilisant des données nouvelles et labellisées de la classe du oc-svm considéré, ou en utilisant les données d'apprentissage des autres classes. On cherche alors à avoir le meilleur rejet possibles sur tous les échantillons disponibles. En effet, l'optimisation est difficile pour un OC-SVM parce que plusieurs séparations sont possibles dans l'espace des descripteurs.

Concernant l'opérateur LBP, le seuil T est fixé à la même valeur que celui utilisé dans le chapitre 2. Des tests ont été réalisés avec un seuil T = 15. Nous avons comparé la première partie du système avec un OC-SVM construit avec différents descripteurs. Ces derniers sont parfois issus de réseau de neurones comme VGG-16 ou GoogleNet [Szegedy15]. Nous avons gardé la dernière couche de GoogleNet avant la couche de décision comme vecteur descripteurs (vecteur de 256 valeurs pour nos patchs) puis injecté dans le OC-SVM. Les poids utilisés par GoogleNet sont ceux du réseaux entraînés sur la base de données ImageNet.

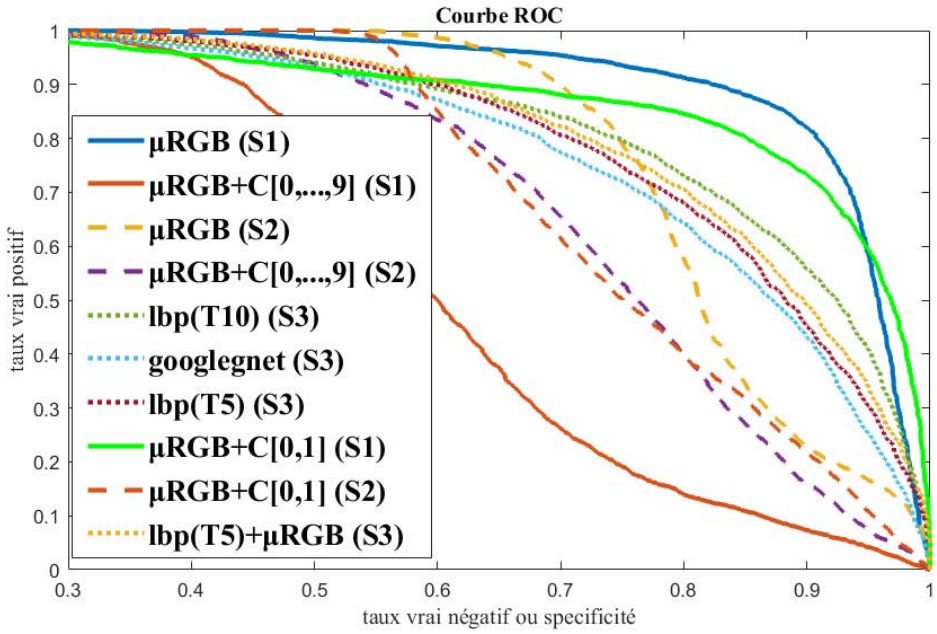
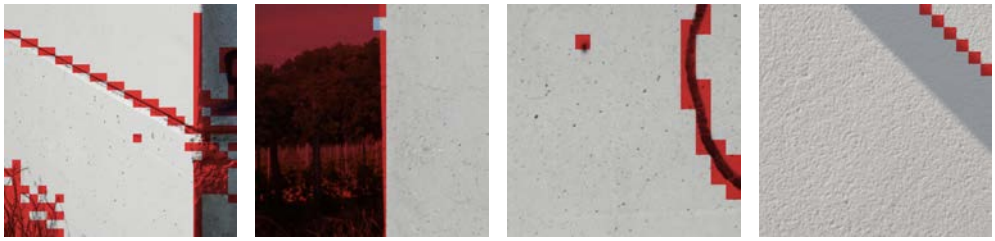


FIGURE 5.5 – Courbe ROC pour la détection d'anomalies en utilisant notre discriminateur et les OC-SVMs selon les scénarios S1, S2 et S3.



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

FIGURE 5.6 – Exemples de détection de défauts par cGAN (z=μRGB) sur des images entières (a) Bonne détection de végétation, graffiti, éclatement et élément de structure (b) rejet du fond (c) Détection de graffiti (d) Problème d'ombre

5.4.4 Résultats

5.4.4.1 Résultats de la détection de défauts

La figure 5.5 nous montre les différentes courbes obtenues dans des configurations différentes. Les OC-SVMs sont comparés avec différents descripteurs et la distance à

l'hyperplan sert de seuil de décision pour construire la courbe. Différentes stratégies avec le cGan ont été adoptées. Le profil LBP suggère de prendre en considération certains types de valeur LBP. Nous avons considéré les sorties du générateur comme dans [Sabokrou18]. Sur les figures 5.5 et 5.7, l'extension "(Sn)" indique le scénario utilisé avec le descripteur associé. Contrairement à [Sabokrou18], la version incluant le générateur en entrée du discriminateur ne donne pas les meilleurs résultats. Le générateur déforme très peu les échantillons appartenant à la classe "anormale" car les images restent assez homogènes dans l'ensemble. Il aurait plutôt pour effet de flouter plus la version anormale. Donner trop d'informations au cGAN n'est pas une bonne stratégie. Les échantillons négatifs (n'appartenant pas à la classe visée) ont dans notre cas des textures assez différentes. C'est pourquoi, les OC-SVMs avec des descripteurs LBP sont assez performants. Cependant, si l'on considère la couleur, il est plus difficile de différencier un graffiti d'un mur normal. En d'autres termes, ajouter une information colorimétrique n'est pas suffisant pour classifier tous les types de défauts.

Dans l'ensemble, les meilleurs résultats sont obtenus avec la moyenne RGB. Cependant, dans la figure 5.7, les tests effectués avec quelques images d'ombres montrent qu'ajouter une information indépendante de l'illumination (comme les descripteurs LBP), améliore la robustesse significativement. Les couches de LBP utilisées sont C^0_{lbp}, C^1_{lbp}.

Pour conclure, les OC-SVMs sont globalement moins performants que le cGAN sur la même base d'apprentissage pour l'ensemble des descripteurs utilisés. Le changement des descripteurs en entrée ne permet pas une nette amélioration des résultats. Cependant, les OC-SVMs restent plus performants que le cGAN sur la combinaison de tous les descripteurs (moyenne RGB et toutes les familles de LBP). En réalité, les deux systèmes n'utilisent pas l'information de la même manière et par conséquent, le choix des descripteurs est plutôt dépendant de la stratégie adoptée. En effet, ajouter des descripteurs pertinents aux OC-SVMs lui permettra de s'améliorer. Ajouter ces mêmes descripteurs au cGAN peut inversement diminuer ses performances. La qualité du discriminateur (et donc du cGAN) va dépendre de la capacité du générateur à fournir

des images réalistes, et ce, tout le long de l'apprentissage. Des descripteurs permettant une génération proche d'une image réaliste ne vont pas améliorer le discriminateur, et celui-ci n'améliorera pas sa capacité à discerner un patch faux d'un vrai. A contrario, une combinaison plus simple de descripteurs, c'est-à-dire avec un peu moins d'informations, permettra de rendre le discriminateur plus performant car son apprentissage sera progressif (comme celui du générateur) et plus robuste. Par conséquent, le choix des descripteurs dépend de la stratégie adoptée. Le cGAN peut être utilisé pour la détection d'anomalies (au sens large), pour améliorer les résultats sous conditions d'avoir des descripteurs adaptés. Dans nos expérimentations, il obtient, toutes combinaisons confondues, les meilleurs résultats. À l'inverse, un mauvais choix de descripteurs peut entraîner une diminution nette des résultats. Si le choix des descripteurs est très délicat, une solution "pratique" serait de se tourner vers les OC-SVMs plus stables à l'ajout de descripteurs mixtes (texture et couleur).

Nous avons comparé notre méthode avec le réseau décrit dans [Oza19] en terme d'accuracy. Ce réseau a été entrainé avec les mêmes données présentées à notre cGAN. Nous obtenons une meilleure accuracy avec 0.82 alors que le modèle de [Oza19] a obtenu 0.69. La modélisation que les auteurs font de la classe négative peut être complexifiée pour notre application.

5.4.4.2 Classification

Methodes	Accuracy (2 classes)
Multi class	.94
Multi One-Class	.91

TABLE 5.3 – Résultats de la classification binaire.

Notre phase de classification est confrontée à un SVM multi-classe standard. Les hyperparamètres sont aussi optimisés avec l'algorithm Grid Search et nous avons gardé les mêmes descripteurs pour les 2 systèmes. Le tableau 5.4.4.2 montre que le système multi-classe est meilleur sans surprise. Dans le cas d'une classification binaire, le multi-classe SVM peut séparer efficacement les classes car le coût de la séparation est cal-

culé en fonction des deux classes d'échantillons. Mais avec plus de données et plus de classes, la performance décroit généralement.

Le système dynamique a besoin de plus de recherche à cause du manque de données. Mais il est prêt à recevoir de nouvelles données. De plus, le système permet de s'adapter à plusieurs types d'infrastructures comme les infrastructures métalliques. L'architecture de reconnaissance de défauts surfaciques est basée sur un framework re-

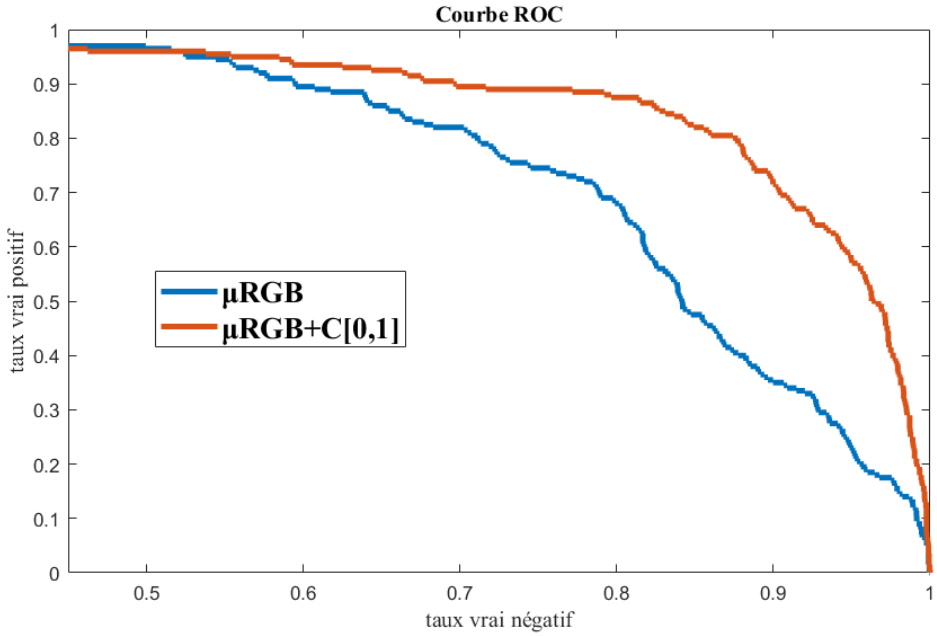


FIGURE 5.7 – Robustesse aux zones d'ombres

présenté à la figure 5.2. Notre architecture est similaire à une classification hiérarchique. En plus d'utiliser les convolutions d'images au sein du GAN initial, elle exploite également les descripteurs LBP définis au chapitre 2.4 ainsi qu'une série de classifieurs one-class SVM.

5.4.5 Détection de défauts sur de grandes images

La détection et classification effectuées ici est réduite à une zone de 64x64 pixels. Réaliser la détection sur une grande image revient donc à réaliser la détection sur plu-

sieurs patchs issus de cette image. Les patchs ainsi détectés/classifiés peuvent faire l'objet d'une mise en relation. Par exemple, un patch identifié comme "sans défaut" au milieu de patchs détectés "défaut" suppose qu'il y a eu une mauvaise détection. La probabilité pour qu'un tel patch soit réellement sans défaut devient alors très faible. C'est pourquoi, il est possible de réaliser d'autres opérations supplémentaires en analysant la disposition des patchs détectés pour améliorer la performance de la détection de défauts en pratique. Quelques cas de figures liés au domaine de l'infrastructure nous permet d'identifier des post-traitements à réaliser une fois la détection des patchs réalisés sur une image :

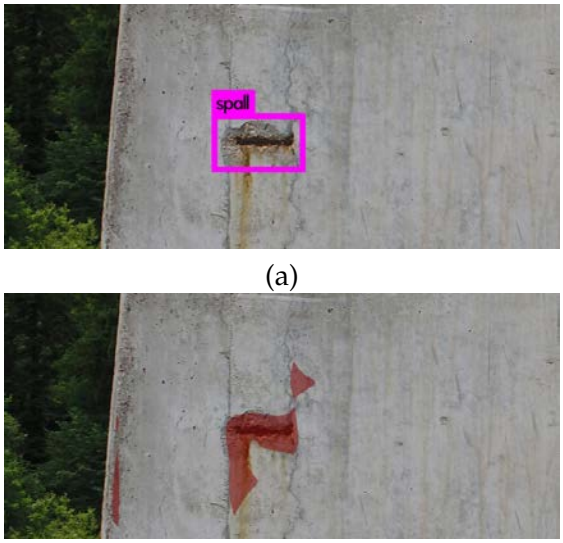
— Plusieurs patchs détectés comme "défauts" disposés sur une ligne peuvent être un élément de structure comme les jonctions de murs.
— Comme l'exemple cité ci-dessus, un patch sans défaut au milieu de patchs considérés comme "défaut" sera, à plus forte probabilité, un patch "défaut" également.
— Les défauts pouvant être très petits, un seul patch isolé ne signifie pas une probabilité moindre que ce dernier soit "sans défaut". Ainsi, aucun patch isolé ne sera supprimé.

Ces traitements sur la disposition des patchs permettent de compléter la détection/classification des défauts sur de grandes images. Des exemples sont montrés sur la figure 5.6.

5.5 Spécialisation sur les éclatements de béton

5.5.1 Apprentissage profond par transfert(*transfer learning*)

Plusieurs articles ont très récemment proposé des architectures différentes pour la détection des éclatements de béton. Très peu d'auteurs se comparent car les bases de données sont souvent privées et les tests effectués présentent les résultats uniquement sur leurs bases. Plusieurs niveaux de détection sont possibles. Dans un premier temps, nous pouvons détecter uniquement la présence ou pas du défaut sur une image par une décision binaire. Les réseaux peuvent être plus complexes et nous pouvons également



(a)

(b)

FIGURE 5.8 – (a) Détection par un réseau type RCNN (YOLO) (b) Détection (et segmentation par un FCN.

localiser les défauts comme dans les R-CNNs. Dans plusieurs articles, nous retrouvons des architectures du même type pour la détection des éclatements. On a testé le réseau YOLO [Redmon16],(voir figure 5.8).

Néanmoins, les experts métiers caractérisent le défaut par son étendue. La connaissance de l'aire occupée par l'éclatement est donc une information dont nous avons besoin. Les réseaux *FCN (Full Convolutional Network)* permettent de réaliser une segmentation sur plusieurs classes de données. La contre partie est qu'il nécessite des données labellisées au niveau pixel. La tâche est alors très coûteuse en temps. Les techniques de *transfer learning* permettent de s'affranchir d'une grande quantité de données. Cependant ces réseaux donnent de bons résultats sur notre jeu de tests. (voir partie expérimentations)
Pour toutes ces raisons, nous avons choisi d'effectuer la segmentation à partir de ces réseaux.

Le *transfer learning* est une technique de deep learning permettant de réaliser un apprentissage sans partir de zéro. On réutilise des connaissances apprises dans d'autres domaines, puis on s'adapte au domaine visé. Lorsqu'il s'agit d'utiliser les poids calcu-

lés d'un réseau de neurones, deux stratégies principales sont possibles. D'une part, les poids peuvent tous évoluer, on transforme alors complètement le domaine d'application sans repartir de zéro. D'autre part, on peut faire évoluer une partie de ces poids. On fixe alors les couches du réseau qui évolueront ou pas. Souvent, les dernières couches qui sont généralement spécialisées dans la tâche voulue sont des couches qui évoluent lors de l'apprentissage. Les réseaux sont donc entraînés avec nos données à partir des poids déjà appris sur une base de donnée différente.

5.5.2 Introduction d'informations supplémentaires dans un réseau

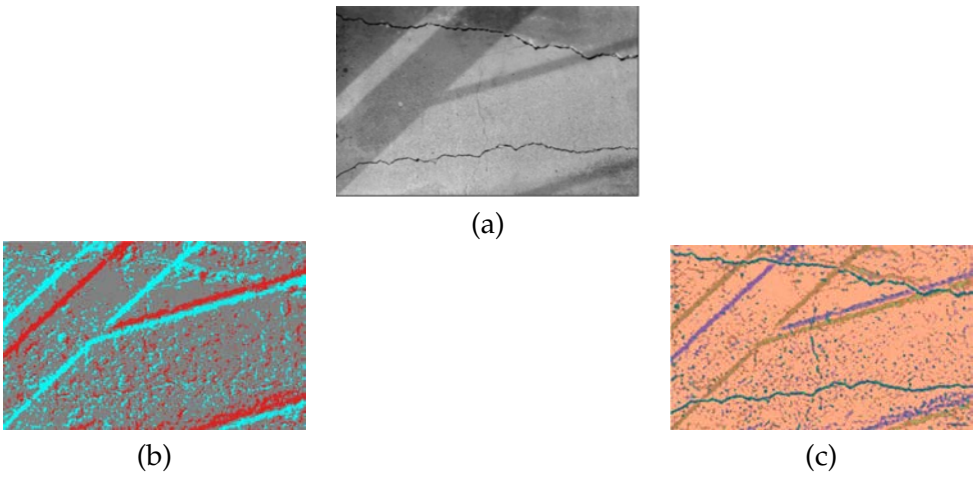
Nous avons cherché à introduire l'information LBP "issu de FLASH" au sein du réseau de neurones pour essayer d'améliorer les résultats. Cependant, l'introduction de l'information est différente de ce que nous avons vu précédemment. Ici, les codes LBP sont transmis directement aux réseaux dans les couches de convolution. Le point problématique est que l'espace des valeurs LBP n'est pas une espace vectoriel euclidien et que l'on ne peut pas appliquer des convolutions. Il faut donc opérer une transformation de cette espace vers un espace euclidien. Plusieurs méthodes sont possibles.

Nous nous sommes inspirés des travaux de [Levi15] et de [Gómez17] pour la transformation. Pour transformer les valeurs des LBP, nous devons construire à la main une mesure de dissimilarité. Cette mesure servira ensuite à réaliser la transformation de l'espace des LBP vers un espace euclidien. On peut retrouver les valeurs dans l'espace métrique, on va utiliser la matrice de dissimilarité suivante :

$$\Delta = \begin{bmatrix} \delta_{1,1} & \cdots & \delta_{1,n} \\ \vdots & \ddots & \vdots \\ \delta_{n,1} & \cdots & \delta_{n,n} \end{bmatrix} \tag{5.4}$$

puis l'équivalence est exprimée de la façon suivante :

$$\delta_{i,j} \approx ||V_i - V_j|| = ||TRANSFORM(lbp_i) - TRANSFORM(lbp_j)|| \tag{5.5}$$



(a) (b) (c)

FIGURE 5.9 – (a) Image originale (b) Mapping LBP (MSD) (c) Mapping LBP (NN)

où $TRANSFORM(lbp_i)$ est la fonction qui permet de transformer la i^{me} valeur possible de LBP exprimée en décimal ou en binaire. Dans notre cas, les valeurs possibles de LBP sont comprises entre 0 et 255.

Nous avons exploré deux méthodes avec deux fonctions de similarités différentes. La première méthode utilise un réseau simple similaire à [Gómez17]. La fonction TRANSFORM est un réseau de neurones sur des pixels représentant le code binaire du LBP. On obtient par surapprentissage une transformation du code LBP dans un espace euclidien. Cependant, nous obtenons une perte importante à la fin de l'apprentissage du réseau.

La deuxième méthode est complètement tirée de [Levi15]. La fonction de dissimilarité est calculée avec l'algorithme Earth Mover Distance, et la fonction TRANSFORM est l'algorithme Multi Dimensional Scaling.

Nous obtenons, avec des représentations colorimétriques différentes, les images de la figure 5.9.

Ce cadre de recherche est intéressant mais nous ne sommes pas allés plus loin. On pourrait imaginer des méthodes hybrides liant les deux méthodes et observer l'évolution des résultats.

5.5.3 Expérimentations

La métrique utilisée est *Intersection over Union* (mean IU). Elle est définie de la façon suivante dans [Shelhamer16] : notons n_{ij} le nombre pixels de classe i ayant pour prédiction une appartenance à la classe j, et n_{cl} le nombre de classes alors on a $meanIU = (1/n_{cl}) \sum_i n_{ii}/(t_i + \sum_j n_{ji} - n_{ii})$ avec t_i le nombre total de pixels appartenant à la classe i.

Base de données.

Nous avons utilisé la base de données CSSC [Yang17]. Cette base constitue notre base de tests tandis que ce sont nos jeux de données qui serviront de base d'apprentissage. Notre base de données est composée de 218 images d'une taille de 256x256 pixels obtenues par découpage d'une dizaine d'images de plus grande résolution. Pour augmenter le nombre de données, nous avons découpé les images avec un taux de recouvrement de 50% sur la largeur et la longueur. Nous avons retenu 174 images pour l'apprentissage. Un échantillon de notre base est représenté sur la figure 5.10.

FIGURE 5.10 – Exemples d'images de notre base de données de défauts surfaciques.

Nous avons réalisé le transfert d'apprentissage en laissant évoluer toutes les couches déjà entraînées du réseau de neurones existant. Nous obtenons les résultats présentés dans le tableau 5.4. Nous avons testé différentes composantes pour l'apprentissage du réseau :

— RGB correspondant aux3 canaux colorimétriques Rouge, Vert et Bleu. Les valeurs des intensités sont comprises entre -1 et 1.

— $LBP_{transform}$ correspond à l'utilisation des motifs LBP transformés pour être dans un espace euclidien.

Composante apprentissage	mean IU	accuracy
RGB	0.65	0.79
$LBP_{transform}$	0.61	0.78
$H + LBP_{transform}$	0.61	0.76

TABLE 5.4 – Comparaison des différents apprentissages de FCN. La base CSSC est utilisé pour les tests.

— $H + LBP_{transform}$ correspond à l'utilisation des composantes précédentes. Nous ajoutons en plus une composante colorimétrique qui est le canal H (hue) de l'espace de couleurs HSV. Ces deux composantes sont normalisées afin d'avoir des valeurs comprises entre -1 et 1.

Ces résultats montrent que contrairement au cGAN, on ne parvient pas à obtenir de meilleurs résultats qu'avec uniquement les composantes rouge, vert et bleu des images. La fusion d'une composante colorimétrique avec l'information LBP n'a pas non plus présentée de meilleurs résultats. À noter que les détections ne sont pas localisées sur les mêmes zones. Une étude approfondie est encore nécessaire mais les résultats sont encourageants pour l'emploi des FCNs dans des environnements plus controlés. Par exemple, sur notre petite base de données, nous obtenons près de 98% de bonne zone détectée.

5.6 Conclusion

Bien que les fissures soient le défaut le plus important, tous les autres types de défauts ne sont pas traités dans la littérature actuelle. Une multitude de systèmes existe pour des défauts spécifiques comme l'éclatement de béton. L'utilisation de réseaux de neurones reste le meilleur moyen d'obtenir de bons résultats pour de gros défauts. Cependant, les petits défauts surfaciques sont difficiles à détecter pour un réseau de neurones. C'est pourquoi, nous proposons également des techniques plus classiques liées à l'analyse de texture. Notre système repose sur une analyse par patch qui permet de détecter de petits défauts. Il repose sur une approche multi one-class SVM et sur l'utilisation de la partie discriminante d'un GAN. Les résultats sont très intéressants

et mènent vers une classification des défauts où le multi OC-SVM montre de bonnes performances. Les recherches sur le choix des descripteurs, des hyperparamètres, ou la transformation des valeurs prédites peuvent amener à de meilleurs résultats.

6

Autres contributions

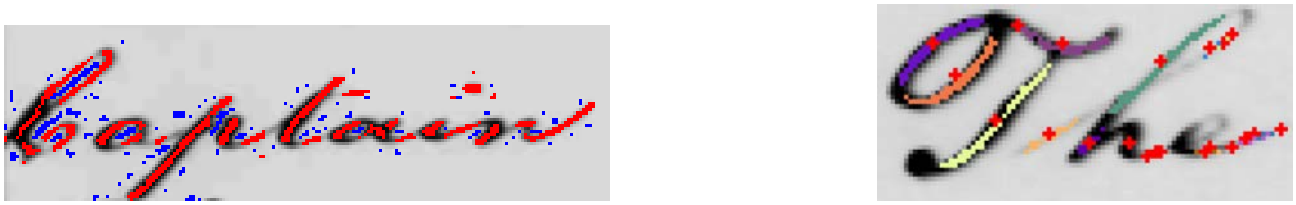
6.1 Introduction

Les points d'intérêt sont à la base de beaucoup d'analyses en traitement d'images. Ils permettent une multitude de taches où les réseaux de neurones ne sont encore pas adaptés. Le suivi d'objets, la reconnaissance de forme, la segmentation etc. sont des exemples où les points d'intérêt sont fondamentaux et antérieurs à toute analyse. Dans ce contexte, on comprend bien que les points d'intérêts FLASH et leurs caractéristiques d'orientation et de contraste peuvent servir dans différents domaines d'application.

Dans ce chapitre, nous verrons que l'utilisation de nos points FLASH peut convenir dans l'analyse de documents. L'intérêt réside dans la vitesse de traitement associée à une utilisation sans apprentissage. Nous présenterons également une utilisation dans la reconnaissance de formes et dans l'aide à la navigation des véhicules intelligents.

6.2 Word Spotting

Beaucoup de bibliothèques et de musées contiennent une énorme quantité de documents manuscrits historiques qui sont digitalisés. Ces documents intéressent beaucoup de personnes, surtout les experts en littérature, les historiens... Malheureusement, la plupart des collections, bien que disponibles auprès les bibliothèques publiques ou sur internet, sont très mal organisées, peu non annotées ou mal indexées. La transcription manuelle de document est une tâche très coûteuse et par conséquent des méthodes automatiques pour construire des documents indexés sont préférées. Cependant, un accès efficace aux collections de documents de manière automatique implique que le système conçu aille au delà de beaucoup de difficultés. Le *word spotting* a été proposé dans le but de permettre un accès aux contenus des documents en retrouvant des mots clés par des requêtes images sans avoir à retranscrire le corpus de documents en entier. Bien que le *word spotting* sur des documents manuscrits soit un domaine de recherche très actif dans l'analyse de documents sur ces deux dernières décennies, il n'y a toujours pas d'approches entièrement satisfaisantes. Comparées aux applications en ligne,



(a) (b)

FIGURE 6.1 – Détection de points FLASH sur des mots. (a) En bleu les points W/B et en rouge les points B/W. (b) Les points clés retenus sont les points rouges étant le barycentre d'un graphe (ensemble de même couleur)

les approches de *word spotting* hors ligne sont considérées comme plus difficiles car on n'a pas d'informations temporelles sur le tracé des caractères à exploiter. De plus, les difficultés sont nombreuses pour les documents manuscrits et historiques : variations des scripteurs, compositions de langage qui divergent, vocabulaire dans des environnements sans contrainte, dégradations multiples et bruit artificiel etc.

6.2.1 Détecteur FLASH

Les points FLASH sont bien adaptés pour les documents manuscrits. Les lignes formées par un scripteur peuvent être détectées par les points FLASH en utilisant nos trois types de points. Plusieurs exemples de détections sont présentés sur la figure 6.1.

Plusieurs points peuvent être détectés sur un même tracé. Par exemple, le tracé de la barre d'un "p" possède plusieurs points. Nous établissons la liaison entre les points en liant les points ayant la même direction et étant très proches. Ensuite, de ces petits graphes, tout comme les fissures, nous pouvons calculer un barycentre qui servira de point d'intérêt dans l'étape suivante et sera représentatif de la partie de lettre écrite.

6.2.2 Descripteurs

L'extraction des points est la première étape dans le processus de reconnaissance de mots-clés. La seconde étape est le calcul d'un descripteur pour chaque point d'intérêt, qui décrira le voisinage. Nous proposons 3 descripteurs autour de nos points FLASH.

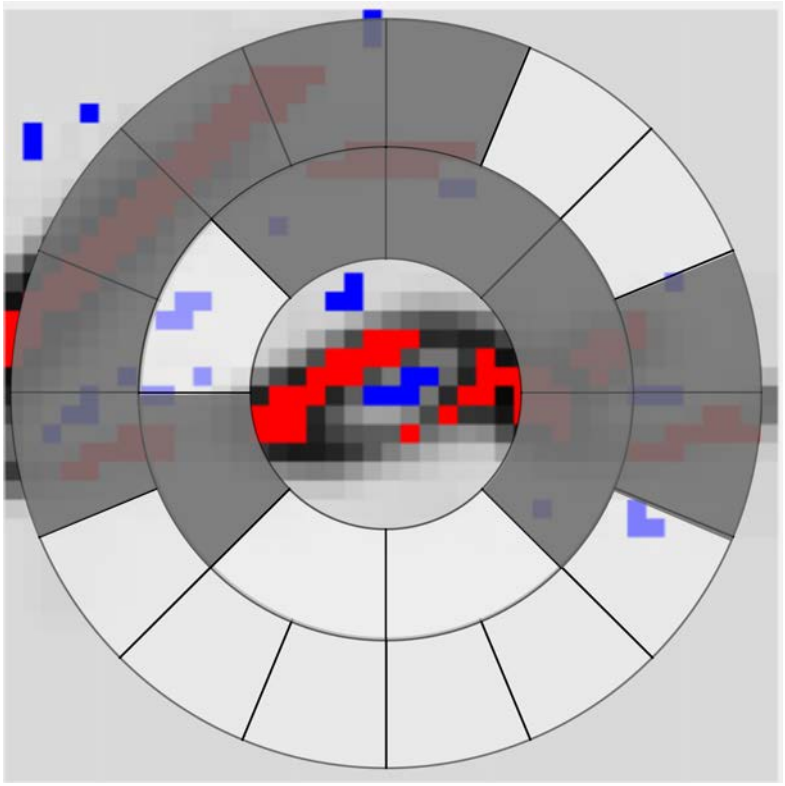


FIGURE 6.2 – Masque circulaire pour construire le descripteur associé à un point FLASH.

Le premier est directement lié à notre descripteur décrit dans le chapitre 4. Le deuxième est plus spécifique pour le *word spotting* et présente de bons résultats sur les bases de tests disponibles. Enfin, le dernier est inspiré de méthodes existantes et présente l'avantage d'être très rapide.

6.2.2.1 Descripteur basé FLASH

Notre premier descripteur est basé sur la division du voisinage d'un point d'intérêt FLASH en plusieurs secteurs. Le découpage est fait en suivant un masque circulaire présenté sur la figure 6.2. Inspiré par notre descripteur destiné à l'appariement d'images de surface de béton, nous avons 2 cercles concentriques divisés en 8 quadrants et 16 quadrants respectivement dans la figure. Dans les expérimentations, le premier cercle à partir du point FLASH possède 16 quadrants et le deuxième a 32 quadrants.

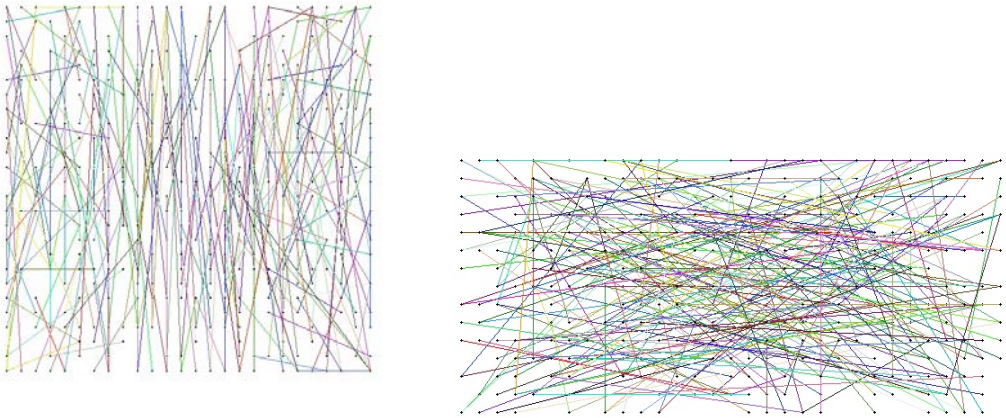
(a) (b)

FIGURE 6.3 – Représentation colorimétrique des valeurs de LBP avec des seuils différents. (a) T=0 (b) T=10 avec le découpage en grille 3x3.

Pour des raisons de simplicité, la figure 6.2 présente un exemple avec 8 quadrants sur le premier cercle et 16 pour le deuxième. Pour chaque quadrant, nous allons chercher d'autres points FLASH sur l'image. Si le point FLASH central est un point blanc sur fond noir, noté point B/W, nous cherchons spécifiquement des points noirs sur fond blanc (noté point W/B). De même, si le point FLASH central est un point W/B, nous cherchons uniquement des points B/W. Nous construisons un descripteur binaire ou chaque quadrant est représenté par un bit. Par conséquent, le descripteur est une chaîne binaire de 48 bits. Si un quadrant contient un point du type recherché, le bit est mis à 1. Sur la figure 6.2, le point central est un point W/B en bleu, et les quadrants assombris contiennent au moins un point rouge (point FLASH B/W), menant à un bit associé égal à 1. Ainsi, une fois le descripteur binaire créé et quelques informations additionnelles ajoutées comme l'orientation du point central, on peut alors déterminer la disposition des points et de retranscrire le voisinage.

6.2.2.2 Descripteur basé LBP

Notre deuxième descripteur est basé sur LBP. Nous exploitons le calcul de la détection des points afin de déduire une valeur de LBP. Le seuil utilisé pour la détection permet de réduire le bruit et agit donc comme un filtre passe bas. Toujours dans la même idée, on divise le voisinage du point central en plusieurs régions. Le découpage le plus simple que nous avons testé, est représenté sur la figure 6.3, où l'on a 9 régions autour du point central. Sur chaque région, on va extraire un histogramme LBP. Les his-



(a) (b)

FIGURE 6.4 – Paires utilisées pour construire le descripteur ORB-LBP. (a) Zone classique de taille 31x31. (b) Notre zone redéfinie pour les manuscrits de taille 15x31.

togrammes seront concaténés et vont construire le vecteur descripteur de la zone noté H. Dans l'idée, la méthode est très similaire au descripteur SIFT mais nous bénéficions de la rapidité de calcul de détecteur FLASH. De plus, la réutilisation des résultats pour les tests effectués pour connaitre la valeur LBP permet de gagner en temps de calcul.

Pour la mesure de similarité, nous utilisons la distance de Bray-Curtis déjà utilisée dans des cas similaires. Cette distance est définie par l'équation suivante :

$$D(H_a, H_b) = \sum_i \frac{|H_a(i) - H_b(i)|}{H_a(i) + H_b(i)} \tag{6.1}$$

où $H_a(i)$ et $H_b(i)$ sont respectivement les $i^{\text{ème}}$ éléments des histogrammes H_a et H_b des points a et b.

6.2.2.3 Descripteur inspiré de ORB : ORB-LBP

Le dernier descripteur que nous proposons est très rapide en temps de calcul. Nous avons repris l'idée de ORB décrit dans le chapitre 4 pour construire un descripteur binaire. Les paires conçues sont adaptées au document. Nous faisons l'hypothèse qu'un document est présenté dans le bon sens sur l'image. Ainsi nous pouvons comparer les

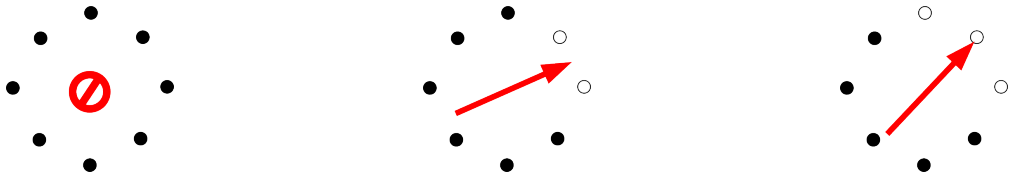


FIGURE 6.5 – (a) Aucune orientation. (b) et (c) ont des orientations similaires

orientations des points dans le voisinage d'un point central. Nous étudions 256 paires dans une zone de 15x31. La zone est réduite verticalement car on ne veut pas prendre en compte des mots des lignes suivantes ou précédentes. La comparaison des points d'une paire donne lieu, comme nous le verrons ci-dessous, à l'assignation de 2 bits de la chaîne binaire. Par conséquent, la chaîne est constituée de 512 bits pour 256 paires de points. Nous avons 2 catégories de code LBP. Ceux qui possèdent une orientation calculable et 2 codes LBP (0 et 255 en décimal) qui n'ont pas d'orientation (voir figure 6.5). On va fixer les 2 bits d'une comparaison de la façon suivante :

— si 2 points n'ont pas les mêmes catégories de code LBP alors les 2 bits sont fixés à 1.

— si 2 points ont la même catégorie de code LBP, alors le premier bit est fixé à 0. Puis le deuxième bit est fixé à 1 si les orientations sont similaires, 0 sinon.

La mesure de similarité est réalisée par l'intermédiaire d'une adaptation de la distance de Hamming proposée par Opencv. Elle se calcule comme la norme de Hamming en considérant la somme de deux bits consécutifs dans la chaine binaire comme un seul bit.

6.2.3 Correspondance de formes par accumulation

Toutes ces méthodes sont employées dans un contexte de recherche de mots. sans segmentation. Nous traitons directement les images d'origine des documents à partir de la requête image. C'est pourquoi après la correspondance des points individuellement, il est primordial de vérifier la cohérence spatiale de l'ensemble du mot. Par

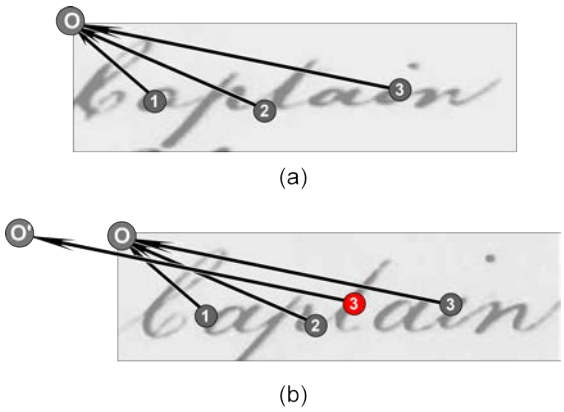


FIGURE 6.6 – Analyse de la cohérence spatiale entre les points correspondants. (a) Requête (b) Bloc candidat à la comparaison issu du document. 4 points correspondent mais 3 seulement sonts corrects.

exemple, en considérant deux anagrammes, ils peuvent avoir les mêmes similarités pour chaque point FLASH mais la disposition spatiale ne sera pas la même.

Tout comme nous l'avons fait pour le problème de l'appariement entre images à partir des points FLASH, l'accumulateur cumule le nombre de points FLASH correspondant et ayant un vecteur identique à celui induit par la requête. Le vecteur est formé par le point clé considéré avec le point origine de la requête. Un exemple d'analyse de la cohérence spatiale est donné dans la figure 6.6. Dans cet exemple, le vecteur induit avec une correspondance avec le point "3" est reporté sur le bloc candidat. Si une quantité suffisante de points obtiennent des vecteurs arrivant sur un même point origine, alors il se peut que nous ayons trouvé une bonne correspondance avec la requête.

6.2.4 Expérimentations

Nous avons testé les différents algorithmes sur la base de Georges Washington. Cette dernière est composée d'un ensemble de lettres manuscrites écrites au 18 siècle par Georges Washington. Les images d'origine sont utilisées sans aucun pré-traitement. La comparaison se fait sur la précision, le rappel et surtout le *mean average precision*

mAP_n. On le définit de la façon suivante :

$$mAP_n = \frac{\sum_{i=1}^{n} P@i \times r(i)}{|rel|} \tag{6.2}$$

$P@n$ est la précision au rang n et est obtenu en considérant la précision calculée uniquement sur les n meilleurs résultats retournés par le système.

Nous avons comparé en suivant la méthodologie originalement proposée par [Leydier09] et repris dans [Liang12]. 38 mots-requêtes représentant un total de 268 requêtes images sont testées. Il s'agit d'une méthode sans segmentation et donc, pour valider un mot trouvé, 50% de la zone du mot clé recherché doit s'intersecter avec la zone retenue par notre détecteur.

Le tableau 6.1 montre les résultats de notre proposition de *word spotting* avec d'autres méthodes de l'état de l'art sans apprentissage. Nos résultats sont similaires aux derniers résultats de l'état de l'art. La méthode dans [Rusiñol15] est basée sur les sacs de mots visuels faites sur le descripteur SIFT. Nous obtenons de meilleurs résultats que la méthode de [Liang12] basée sur une segmentation préalable des mots. Avec la base de Georges Washington, l'utilisation directe de l'algorithme LBP obtient un meilleur résultat que la méthode avec histogramme. Notre méthode fusionnant ORB-LBP est plus rapide que celle utilisant les histogrammes LBP et est plus performante. Cependant, dans la théorie, elle semble assez sensible à la rotation ou un changement d'écriture (par exemple passage de l'italique à droit). Elle est actuellement plus similaire à BRIEF que ORB mais la comparaison de la chaine binaire est inspirée de ORB.

Nous avons proposé une solution pour le *word spotting* en abordant que des méthodes non neuronales exploitant les scénarios graphomorphologiques dans l'analyse des mots (points d'intérêt et sac de mots visuels).

Méthodes	mAP_{10}
Histogramme LBP	0.59
ORB (classique)	0.69
ORB-LBP	0.78
[Rusiñol15]	0.79
[Liang12](avec segmentation)	0.67

TABLE 6.1 – Résultats du word spotting suivant la procédure et la métrique de [Leydier09]

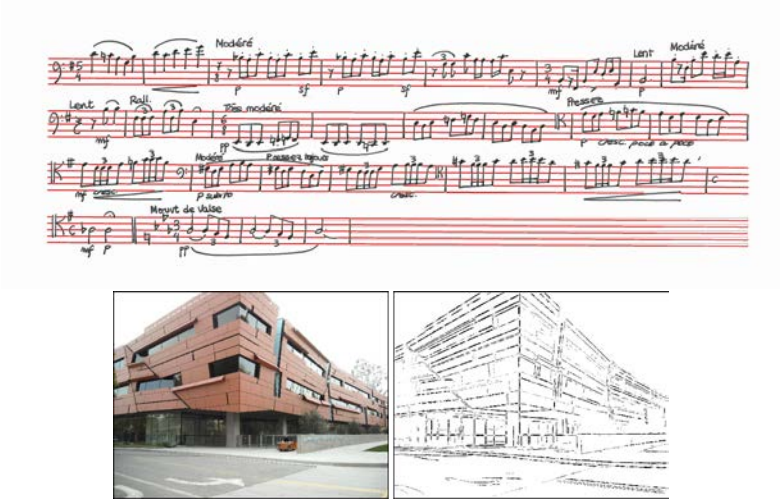
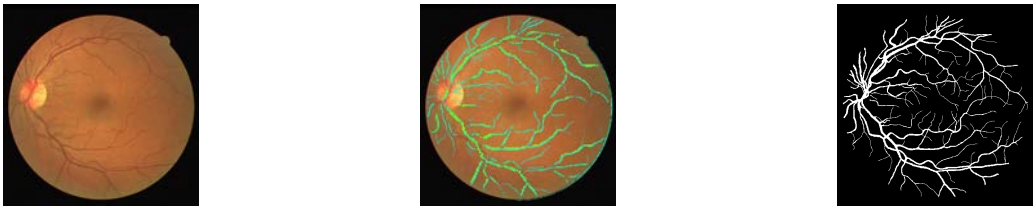


FIGURE 6.7 – Exemples de détection de lignes droites

6.3 Détection de lignes droites et de segments orientés

La mise en correspondance des points FLASH (spécifiquement les micro-lignes) peut être étendue en contraignant la relation entre les points. Si on impose un angle plus réduit entre des points voisins alors nous pouvons détecter des lignes droites. A l'aide d'un filtre de Sobel, nous pouvons alors détecter des contours sur n'importe quel type d'image. Deux exemples sont présentés sur la figure 6.7.

Nous avons également effectué des tests dans d'autres domaines d'applications et utilisant la détection des points FLASH. Dans ces applications, la détection est réalisée à plusieurs échelles afin d'obtenir une détection de traits épais sur une image. Le temps de calcul de ces images est très rapide car la résolution des images est souvent



(a) (b) (c)

FIGURE 6.8 – Exemples de détection des vaisseaux sanguins extrait de la base DRIVE. (a) Image original (b) Détection multi échelle par FLASH (c) Vérité terrain

FIGURE 6.9 – Exemples de détection FLASH sur la base de données LISA

très basse dans les bases de données utilisées. Nous n'avons pas fait d'analyse comparative et quantitative entre des méthodes existantes dans ces domaines d'applications. Nous montrons à titre de perspectives d'application, soulignant le potentiel pour améliorer les temps de traitement des algorithmes existants. Les figures 6.8 et 6.9 montrent quelques exemples pour différentes applications sur des images issues d'un contexte mobile (caméra embarquée dans une voiture) et des images de laboratoire médicaux (vaisseaux sanguins dans la rétine).

6.4 Conclusion

La détection des points d'intérêts FLASH a permis des applications liées à notre objectif de départ mais aussi à d'autres domaines où une détection de lignes ou de segments orientés est nécessaire. Ces applications démontrent la polyvalence de nos points FLASH. L'algorithme d'appariement des images décrit dans le chapitre 4 a inspiré les algorithmes de *word spotting*. Les résultats obtenus sont encourageants et peuvent être

améliorés dans le contexte sans segmentation et sans apprentissage. La détection des fissures peut aussi être directement appliquée à la détection de lignes droites simplement en modifiant quelques paramètres.

7

Conclusion et Perspectives

Nous allons conclure sur les travaux réalisés en reprenant les 4 grandes étapes du processus global rattachées aux objectifs initiaux. L'évolution au sein de l'entreprise Smart Aerial Machines a également orienté la réflexion. Les travaux menés n'ont pas suivi le processus global logique d'acquisition. L'ordre des chapitres suit à peu près le cheminement de la réflexion autour de ce sujet de thèse. En reprenant l'ordre logique d'acquisition nous synthétisons ici les contributions apportées et les perspectives à venir.

DÉTERMINATION DE LA QUALITÉ DES IMAGES.

L'acquisition sera validée si après cette dernière, l'opérateur n'a pas besoin de revenir sur le terrain pour prendre des images de parties manquantes. L'objectif est donc d'assurer la qualité d'une image prise par un vecteur d'acquisition au moment de l'acquisition.

Nous nous sommes focalisés sur le flou et avons proposé une métrique basée sur la quantité de flou pour noter la qualité d'une image. L'algorithme est rapide en temps de calcul (quasi temps réel). Il est calculé en même temps que plusieurs métriques et permet une économie en ressources. Une analyse des motifs binaires locaux est réalisée et leur décompte conduit à une valeur qui détermine la quantité de flou dans l'image.

L'objectif est atteint pour une partie des dégradations que peut subir une image lors de l'acquisition. Cependant, il reste toujours à déterminer comment s'assurer que les autres types de dégradations (bruit, éblouissement...) n'empêchent pas le traitement des données.

Des pistes d'améliorations sont envisagées pour s'adapter aux types de murs. Rendre notre métrique polyvalente permettra de s'adapter à plusieurs types de murs comme la maçonnerie ou les ouvrages en fer. Le temps de calcul est encore améliorable en exploitant des techniques de traitement GPU. En effet, le calcul du code LBP peut être totalement parallélisé.

DÉTECTION DES FISSURES.

Le défaut le plus important est la fissure sur les ouvrages en béton. La détection temps réel est nécessaire afin de savoir s'il faut reprendre une photographie à plus haute réso-

lution ou pas. Nous avons développé un détecteur, nommé FLASH (*Fast Local Analysis by threSHolding*), qui se base sur une analyse locale des pixels qui a donné lieu à une première publication [Faula18]. Les résultats montrent que la détection des fissures permet d'avoir peu de fausses alarmes. La détection sur toute la longueur de la fissure n'est pas optimale car parfois plusieurs morceaux de fissures sont détectés.

Ici aussi, nous pouvons exploiter le traitement hautement parallèle. La gestion de la structure de données "image" optimisée pour la mise en relation des points ne sera pas triviale. Cependant, on ne note pas de verrou pour ce sujet. Ensuite, nous n'avons pas réalisé de traitement multi-échelle sur les surfaces de béton car on considère qu'une première photographie de qualité suffisante permet de caractériser une fissure (ouverture, longueur, type...).

Par la suite, les perspectives de réaliser et de fusionner un traitement multi-échelle sont à envisager pour améliorer la détection. Une autre amélioration est possible si on supprime les contraintes de temps. En exploitant, le potentiel des réseaux de neurones, nos points FLASH peuvent être intégrés dans un réseau pour aider à la détection. En effet, à ce jour, seules les grosses fissures sont détectées par les réseaux à convolution.

APPARIEMENT DES IMAGES.

Après avoir détecté les fissures, premier défaut critique, il a fallu s'assurer d'abord qu'on ne détectait pas plusieurs fois le même défaut sur différentes images. Tel a été le premier objectif de l'appariement qui nous a conduit à un descripteur très performant en présence d'une fissure.
Un deuxième objectif est apparu par la suite : s'assurer que le recouvrement imposé était respecté. Le même algorithme proposé sert à calculer un recouvrement dans la limite de notre méthode, c'est-à-dire avec un changement d'échelle réduit, des mouvements essentiellement composés de translation sur des murs peu texturés.

Une piste d'améliorations consiste à considérer non pas un cercle pour la construction de la mise en correspondance entre signatures mais des ellipses de ratio grand axe/petit axe variable. Nous pouvons alors être en théorie totalement invariant aux changements de point de vue. Cependant, le temps de calcul sera décuplé et on aura

la présence de plus de confusion. Il s'agit d'une voie intéressante à explorer afin de déterminer directement la différence de point de vue entre deux prises de vue.

DÉTECTION DES DÉFAUTS SURFACIQUES.
Une fois l'acquisition assurée, on peut en post traitement effectuer un traitement sur d'autres types de défauts plus faciles à identifier car plus gros. Les techniques de *transfer learning* permettent d'avoir une segmentation des défauts si on dispose d'assez de données pour l'apprentissage des réseaux.

Au vue du peu de données disponibles sur les différents types de défauts, notre contribution consiste en un système capable d'évoluer dans le temps et permet de détecter tout type de défauts surfaciques sur une surface de béton. La classification est limitée aux éclatements et aux graffitis.
Des perspectives sont envisagées tout d'abord dans la complétion des bases de données pour augmenter le nombre de défauts à classifier. L'analyse se fait actuellement sur des patchs découpés depuis une image. Nous pouvons améliorer les probabilités de détection en exploitant des relations spatiales des patchs détectés comme défaut.

Au cours de cette thèse, nous nous sommes focalisés sur la reconnaissance de défauts sur des ouvrages en béton. Ces derniers sont les plus contraignants pour la détection des défauts critiques comme les fissures. Cependant, un large éventail d'ouvrages est possible. On retrouve les ouvrages en maçonnerie, très nombreux en France, ou les ouvrages en fer. Chacun a des spécificités propres mais également des points communs. Par exemple, on recherchera toujours des fissures mais avec une précision moins grande. Les techniques peuvent être adaptées de différentes manières. Quelques pistes citées ici serviront à un travail plus approfondi :

— la détection de fissures sur des ouvrages en maçonnerie peut se faire avec les mêmes algorithmes mais en réalisant un pré-traitement qui consiterait à isoler/identifer les éléments de maçonnerie.
— la détection des défauts plus grands pourrait être réalisée en exploitant les descripteurs texture pour les ouvrages en maçonnerie. Pour les ouvrages en fer,

une acquisition de quantités de données permettrait d'avoir des résultats intéressants. En effet, l'obstacle principale est le manque de données. Cependant, l'apprentissage d'un ou plusieurs réseaux dédiés à la détection sur les différents types d'infrastructures n'est pas une solution optimale pour être utilisée par des opérateurs experts. Ces derniers ne sont ni prêts ni formés à gérer plusieurs modèles d'apprentissage. D'autres solutions plus polyvalentes sont donc à imaginer.

Sur le sujet général de la thèse qui consiste en l'extraction de caractéristiques sur des images acquises en contexte mobile, la mise en place de nos algorithmes et méthodes a permis d'ouvrir le champ vers d'autres applications intéressantes. Par exemple, la détection de lignes droites est utilisée aujourd'hui pour améliorer la fiabilité des algorithmes de SLAM en temps réel. Ainsi, nous pouvons nous positionner comme une alternative aux algorithmes existants pour la première phase de détection de lignes. Nous sommes encouragés à réaliser des tests par la suite sur un plus grand nombre de données.

La plus grande frustration a été de ne pas utiliser réellement nos algorithmes dans un vecteur d'acquisition tel qu'un drone. Cependant, les algorithmes ont été testés sur des ordinateurs embarqués. Le secteur de la surveillance des infrastructures est en plein essor et certaines questions liées à ce sujet de thèse sont encore ouvertes (par exemple le SLAM en temps réel).

La grande précision demandée par les experts a constitué le plus grand défi dans le développement d'algorithmes robustes et cette thèse permet d'apporter une ébauche de solution globale à la surveillance des infrastructures.

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	**M. Stéphane DANIELE** Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	**M. Gérard SCORLETTI** École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES **secretariat.e2m2@univ-lyon1.fr**	**M. Philippe NORMAND** UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Curien - 3ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tel : +33(0)4 72 44 82 32 **sylvie.ricard-blum@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 **infomaths@univ-lyon1.fr**	**M. Hamamache KHEDDOUCI** Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tel : 04.72.44.83.69 **hamamache.kheddouci@univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **ed.materiaux@insa-lyon.fr**	**M. Jean-Yves BUFFIÈRE** INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **mega@insa-lyon.fr**	**M. Jocelyn BONJOUR** INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **veronique.cervantes@univ-lyon2.fr**	**M. Christian MONTES** Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Bibliographie

[Abdel-Qader03] I. Abdel-Qader, O. Abudayyeh & M. Kelly. *Analysis of Edge-Detection Techniques for Crack Identification in Bridges*. Journal of Computing in Civil Engineering, vol. 17, no. 4, pages 255–263, September 2003.

[Andrearczyk17] Vincent Andrearczyk. *Deep learning for texture and dynamic texture analysis*. doctoral, Dublin City University. School of Electronic Engineering, November 2017. Publication Title : Andrearczyk, Vincent (2017) Deep learning for texture and dynamic texture analysis. PhD thesis, Dublin City University.

[Anwar14] Said Amirul Anwar & Mohd Zaid Abdullah. *Micro-crack detection of multicrystalline solar cells featuring an improved anisotropic diffusion filter and image segmentation technique*. EURASIP Journal on Image and Video Processing, vol. 2014, no. 1, page 15, December 2014.

[Bahrami14] K. Bahrami & A. C. Kot. *A Fast Approach for No-Reference Image Sharpness Assessment Based on Maximum Local Variation*. IEEE Signal Processing Letters, vol. 21, no. 6, pages 751–755, June 2014.

[Bay08] Herbert Bay, Andreas Ess, Tinne Tuytelaars & Luc Van Gool. *Speeded-Up Robust Features (SURF)*. Computer Vision and Image Understanding, vol. 110, no. 3, pages 346–359, June 2008.

[Belongie02] S. Belongie, J. Malik & J. Puzicha. *Shape matching and object recognition using shape contexts*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pages 509–522, April 2002.

[Bensoltana14] Wael Bensoltana, A. Porebski, Nicolas Vandenbroucke, Adeel Ahmad & Denis Hamad. *Contribution des descripteurs de texture LBP à la classification d'images de dentelles*. November 2014.

[Calonder12] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha & P. Fua. *BRIEF : Computing a Local Binary Descriptor Very Fast*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 7, pages 1281–1298, July 2012.

[Cha17a] Young-Jin Cha, Wooram Choi & Oral Büyüköztürk. *Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks*. Computer-Aided Civil and Infrastructure Engineering, vol. 32, no. 5, pages 361–378, May 2017.

[Cha17b] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani & Oral Büyüköztürk. *Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types*. Computer-Aided Civil and Infrastructure Engineering, vol. 33, no. 9, pages 731–747, November 2017.

[Chambon10] Sylvie Chambon. *Detection of Road Cracks with Multiple Images*. site perso IRIT.com VISAPP, May 2010. International Joint Conference on Computer Vision Theory and Applications, Angers, FRANCE.

[Chanda14] Sukalpa Chanda, Guoping Bu, Hong Guan, Jun Jo, Umapada Pal, Yew-Chaye Loo & Michael Blumenstein. *Automatic Bridge Crack Detection – A Texture Analysis-Based Approach*. In Neamat El Gayar, Friedhelm Schwenker & Cheng Suen, editeurs, Artificial Neural Networks in Pattern Recognition, numéro 8774 in Lecture Notes in Computer Science, pages 193–203. Springer International Publishing, October 2014.

[Cimpoi14] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed & Andrea Vedaldi. *Describing Textures in the Wild*. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 3606–3613, June 2014. ISSN : 1063-6919.

[Cui14] Fang Cui, Zhe Li & Li Yao. *Images Crack Detection Technology based on Improved K-means Algorithm*. Journal of Multimedia, vol. 9, no. 6, June 2014.

[Daugman85] John G. Daugman. *Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters*. J. Opt. Soc. Am. A, JOSAA, vol. 2, no. 7, pages 1160–1169, July 1985.

[Dorafshan18] Sattar Dorafshan, Robert J. Thomas & Marc Maguire. *Comparison of deep convolutional neural networks and edge detectors for image-*

based crack detection in concrete. Construction and Building Materials, vol. 186, pages 1031–1045, October 2018.

[Duda72] Richard O. Duda & Peter E. Hart. *Use of the Hough Transformation to Detect Lines and Curves in Pictures*. Commun. ACM, vol. 15, no. 1, pages 11–15, January 1972.

[Fan12] Bin Fan, Fuchao Wu & Zhanyi Hu. *Robust line matching through line–point invariants*. Pattern Recognition, vol. 45, no. 2, pages 794–805, February 2012.

[Faula18] Yannick Faula, Stéphane Bres & Véronique Eglin. *FLASH : A New Key Structure Extraction used for Line or Crack Detection*. pages 446–452, April 2018.

[Feichtenhofer13] C. Feichtenhofer, H. Fassold & P. Schallauer. *A Perceptual Image Sharpness Metric Based on Local Edge Gradient Analysis*. IEEE Signal Processing Letters, vol. 20, no. 4, pages 379–382, April 2013.

[Ferzli09] R. Ferzli & L. J. Karam. *A No-Reference Objective Image Sharpness Metric Based on the Notion of Just Noticeable Blur (JNB)*. IEEE Transactions on Image Processing, vol. 18, no. 4, pages 717–728, April 2009.

[Fischler81] Martin A. Fischler & Robert C. Bolles. *Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. Commun. ACM, vol. 24, no. 6, pages 381–395, June 1981.

[Fujita10] Yusuke Fujita & Yoshihiko Hamamoto. *A robust automatic crack detection method from noisy concrete surfaces*. Machine Vision and Applications, vol. 22, no. 2, pages 245–254, February 2010.

[Gao18] Yuqing Gao & Khalid M. Mosalam. *Deep Transfer Learning for Image-Based Structural Damage Recognition*. Computer-Aided Civil and Infrastructure Engineering, vol. 33, no. 9, pages 748–768, September 2018.

[Gavilán11] Miguel Gavilán, David Balcones, Oscar Marcos, David F. Llorca, Miguel A. Sotelo, Ignacio Parra, Manuel Ocaña, Pedro Aliseda, Pedro Yarza & Alejandro Amírola. *Adaptive Road Crack Detection System by Pavement Classification*. Sensors, vol. 11, no. 12, pages 9628–9657, October 2011.

[German12] Stephanie German, Ioannis Brilakis & Reginald DesRoches. *Rapid entropy-based detection and properties measurement of concrete*

spalling with machine vision for post-earthquake safety assessments. Advanced Engineering Informatics, vol. 26, no. 4, pages 846–858, October 2012.

[Gioi12] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel & Gregory Randall. *LSD : a Line Segment Detector*. Image Processing On Line, vol. 2, pages 35–55, March 2012.

[Girshick15] Ross Girshick. *Fast R-CNN*. arXiv :1504.08083 [cs], September 2015. arXiv : 1504.08083.

[Goodfellow14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville & Yoshua Bengio. *Generative Adversarial Networks*. arXiv :1406.2661 [cs, stat], June 2014. arXiv : 1406.2661.

[Gu15] K. Gu, G. Zhai, W. Lin, X. Yang & W. Zhang. *No-Reference Image Sharpness Assessment in Autoregressive Parameter Space*. IEEE Transactions on Image Processing, vol. 24, no. 10, pages 3218–3231, October 2015.

[Gvozden18] Goran Gvozden, Sonja Grgic & Mislav Grgic. *Blind image sharpness assessment based on local contrast map statistics*. Journal of Visual Communication and Image Representation, vol. 50, pages 145–158, January 2018.

[Gómez17] L. Gómez, M. Rusiñol & D. Karatzas. *LSDE : Levenshtein Space Deep Embedding for Query-by-String Word Spotting*. In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 01, pages 499–504, November 2017.

[Harris88] Chris Harris & Mike Stephens. *A combined corner and edge detector*. 1988.

[Hu10] Han Hu, Quanquan Gu & Jie Zhou. *HTF : a novel feature for general crack detection*. In 2010 17th IEEE International Conference on Image Processing (ICIP), pages 1633–1636, September 2010.

[Humeau-Heurtier19] Anne Humeau-Heurtier. *Texture Feature Extraction Methods : A Survey*. IEEE Access, vol. 7, pages 8975–9000, 2019. Conference Name : IEEE Access.

[Isola17] P. Isola, J. Zhu, T. Zhou & A. A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5967–5976, July 2017.

[Ito02] A. Ito, Y. Aoki & S. Hashimoto. *Accurate extraction and measurement of fine cracks from concrete block surface image*. In IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the], volume 3, pages 2202–2207 vol.3, November 2002.

[Iyer06] Shivprakash Iyer & Sunil K. Sinha. *Segmentation of Pipe Images for Crack Detection in Buried Sewers*. Computer-Aided Civil and Infrastructure Engineering, vol. 21, no. 6, pages 395–410, 2006.

[Jahanshahi11] Mohammad R. Jahanshahi, Sami F. Masri, Curtis W. Padgett & Gaurav S. Sukhatme. *An innovative methodology for detection and quantification of cracks through incorporation of depth perception*. Machine Vision and Applications, vol. 24, no. 2, pages 227–241, December 2011.

[Jain04] Deepesh Jain. *Superresolution using Papoulis-Gerchberg Algorithm*, 2004. Digiital Video Processing.

[Kim18] Hyunjun Kim, Eunjong Ahn, Myoungsu Shin & Sung-Han Sim. *Crack and Noncrack Classification from Concrete Surface Images Using Machine Learning*. Structural Health Monitoring, page 1475921718768747, April 2018.

[K.N.SIVABALAN10] K.N.SIVABALAN & Dr.D.GHANADURAI. *Detection of defects in digital texture images using segmentation*. International Journal of Engineering Science and Technology, vol. 2, no. 10, pages 5187–5191, October 2010.

[Koch11] Christian Koch & Ioannis Brilakis. *Pothole detection in asphalt pavement images*. Advanced Engineering Informatics, vol. 25, no. 3, pages 507–515, August 2011.

[Koch15] Christian Koch, Kristina Georgieva, Varun Kasireddy, Burcu Akinci & Paul Fieguth. *A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure*. Advanced Engineering Informatics, vol. 29, no. 2, pages 196–210, April 2015.

[La14] H. M. La, N. Gucunski, Seong-Hoon Kee, J. Yi, T. Senlet & Luan Nguyen. *Autonomous robotic system for bridge deck data collection and analysis*. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1950–1955, September 2014.

[Leutenegger11] S. Leutenegger, M. Chli & R. Y. Siegwart. *BRISK : Binary Robust invariant scalable keypoints*. In 2011 International Conference on Computer Vision, pages 2548–2555, November 2011.

[Levi15] Gil Levi & Tal Hassner. *Emotion Recognition in the Wild via Convolutional Neural Networks and Mapped Binary Patterns*. In Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, ICMI '15, pages 503–510, New York, NY, USA, 2015. ACM.

[Leydier09] Y. Leydier, A. Ouji, F. LeBourgeois & H. Emptoz. *Towards an omnilingual word retrieval system for ancient manuscripts*. Pattern Recognition, vol. 42, no. 9, pages 2089–2105, 2009.

[Li16] L. Li, W. Lin, X. Wang, G. Yang, K. Bahrami & A. C. Kot. *No-Reference Image Blur Assessment Based on Discrete Orthogonal Moments*. IEEE Transactions on Cybernetics, vol. 46, no. 1, pages 39–50, January 2016.

[Li17] Y. Li, Z. Wang, G. Dai, S. Wu, S. Yu & Y. Xie. *Evaluation of realistic blurring image quality by using a shallow convolutional neural network*. In 2017 IEEE International Conference on Information and Automation (ICIA), pages 853–857, July 2017.

[Liang12] Y. Liang, M. C. Fairhurst & R. M. Guest. *A synthesised word approach to word retrieval in handwritten documents*. Pattern Recognition, vol. 45, no. 12, pages 4225–4236, December 2012.

[Lins16] R.G. Lins & S.N. Givigi. *Automatic Crack Detection and Measurement Based on Image Analysis*. IEEE Transactions on Instrumentation and Measurement, vol. PP, no. 99, pages 1–8, 2016.

[Liu16a] Li Liu, Paul Fieguth, Xiaogang Wang, Matti Pietikäinen & Dewen Hu. *Evaluation of LBP and Deep Texture Descriptors with a New Robustness Benchmark*. In Bastian Leibe, Jiri Matas, Nicu Sebe & Max Welling, editeurs, Computer Vision – ECCV 2016, Lecture Notes in Computer Science, pages 69–86, Cham, 2016. Springer International Publishing.

[Liu16b] Li Liu, Paul Fieguth, Guoying Zhao, Matti Pietikäinen & Dewen Hu. *Extended local binary patterns for face recognition*. Information Sciences, vol. 358-359, pages 56–72, September 2016.

[Lowe04] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. Int. J. Comput. Vision, vol. 60, no. 2, pages 91–110, November 2004.

[Mair10] Elmar Mair, Gregory D. Hager, Darius Burschka, Michael Suppa & Gerhard Hirzinger. *Adaptive and Generic Corner Detection Based on the Accelerated Segment Test*. In Kostas Daniilidis, Petros Maragos & Nikos Paragios, editeurs, Computer Vision – ECCV 2010, numéro 6312 in Lecture Notes in Computer Science, pages 183–196. Springer Berlin Heidelberg, September 2010.

[Makantasis15] K. Makantasis, E. Protopapadakis, A. Doulamis, N. Doulamis & C. Loupos. *Deep Convolutional Neural Networks for efficient vision based tunnel inspection*. In 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), pages 335–342, September 2015.

[Mavridaki14] E. Mavridaki & V. Mezaris. *No-reference blur assessment in natural images using Fourier transform and spatial pyramids*. In 2014 IEEE International Conference on Image Processing (ICIP), pages 566–570, October 2014.

[Medina10] Roberto Medina, Jaime Gómez-García-Bermejo, Eduardo Zalama, Roberto Medina, Jaime Gómez-García-Bermejo & Eduardo Zalama. *Automated visual inspection of road surface cracks*. ISARC Proceedings, vol. 2010 Proceedings of the 27th ISARC, Bratislava, Slovakia, pages 155–164, 2010.

[Mikolajczyk05] K. Mikolajczyk & C. Schmid. *A performance evaluation of local descriptors*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pages 1615–1630, October 2005.

[Mirza14] Mehdi Mirza & Simon Osindero. *Conditional Generative Adversarial Nets*. arXiv :1411.1784 [cs, stat], November 2014. arXiv : 1411.1784.

[Mitchell20] A. Mitchell. The Esri Guide to GIS Analysis : Geographic Patterns and Relationships. Volume 1. ESRI Guide to GIS Analysis. Esri Press, 2020.

[Mittal12] A. Mittal, A. K. Moorthy & A. C. Bovik. *No-Reference Image Quality Assessment in the Spatial Domain*. IEEE Transactions on Image Processing, vol. 21, no. 12, pages 4695–4708, December 2012.

[Narvekar09] N. D. Narvekar & L. J. Karam. *A no-reference perceptual image sharpness metric based on a cumulative probability of blur detection*. In 2009 International Workshop on Quality of Multimedia Experience, pages 87–91, July 2009.

[Nasrollahi14] Kamal Nasrollahi & Thomas B. Moeslund. *Super-resolution : a comprehensive survey*. Machine Vision and Applications, vol. 25, no. 6, pages 1423–1468, June 2014.

[Ngo Ho17] Anh Khoi Ngo Ho, Véronique Eglin, Nicolas Ragot & Jean-Yves Ramel. *A multi-one-class dynamic classifier for adaptive digitization of document streams*. International Journal on Document Analysis and Recognition, pages 1–18, May 2017.

[Nguyen10] Tien Sy Nguyen. *Extraction de structures fines sur des images texturées : application à la détection automatique de fissures sur des images de surface de chaussées*. phdthesis, Université d'Orléans, November 2010.

[Nievergelt13] Yves Nievergelt. *Haar's Simple Wavelets*. In Wavelets Made Easy, Modern Birkhäuser Classics, pages 3–35. Springer New York, 2013.

[Ojala02] T. Ojala, M. Pietikainen & T. Maenpaa. *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pages 971–987, July 2002.

[Oliveira13] H. Oliveira & P. L. Correia. *Automatic Road Crack Detection and Characterization*. IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 1, pages 155–168, March 2013.

[Oza19] Poojan Oza & Vishal M. Patel. *One-Class Convolutional Neural Network*. IEEE Signal Process. Lett., vol. 26, no. 2, pages 277–281, February 2019. arXiv : 1901.08688.

[Pereira15] Fábio Celestino Pereira & Carlos Eduardo Pereira. *Embedded Image Processing Systems for Automatic Recognition of Cracks using {UAVs}*. IFAC-PapersOnLine, vol. 48, no. 10, pages 16 – 21, 2015. 2nd {IFAC} Conference on Embedded Systems, Computer Intelligence and Telematics {CESCIT} 2015Maribor, Slovenia, 22-24 June 2015.

[Phung17] Manh Duong Phung, Van Truong Hoang, Tran Hiep Dinh & Quang Ha. *Automatic Crack Detection in Built Infrastructure Using Unmanned Aerial Vehicles*. July 2017.

[Prasanna14] P. Prasanna, K.J. Dana, N. Gucunski, B.B. Basily, H.M. La, R.S. Lim & H. Parvardeh. *Automated Crack Detection on Concrete Bridges*. IEEE Transactions on Automation Science and Engineering, vol. PP, no. 99, pages 1–9, 2014.

[Qu15] Zhong Qu, Li-Dan Lin, Yang Guo & Ning Wang. *An improved algorithm for image crack detection based on percolation model*. IEEJ Transactions on Electrical and Electronic Engineering, vol. 10, no. 2, pages 214–221, 2015.

[Redmon16] Joseph Redmon & Ali Farhadi. *YOLO9000 : Better, Faster, Stronger*. arXiv :1612.08242 [cs], December 2016. arXiv : 1612.08242.

[Ren16] Shaoqing Ren, Kaiming He, Ross Girshick & Jian Sun. *Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks*. arXiv :1506.01497 [cs], January 2016. arXiv : 1506.01497.

[Rublee11] E. Rublee, V. Rabaud, K. Konolige & G. Bradski. *ORB : An efficient alternative to SIFT or SURF*. In 2011 International Conference on Computer Vision, pages 2564–2571, November 2011.

[Rusiñol15] Marçal Rusiñol, David Aldavert, Ricardo Toledo & Josep Lladós. *Efficient segmentation-free keyword spotting in historical document collections*. Pattern Recognition, vol. 48, no. 2, pages 545–555, February 2015.

[Sabokrou18] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy & Ehsan Adeli. *Adversarially Learned One-Class Classifier for Novelty Detection*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[Salman13] M. Salman, S. Mathavan, K. Kamal & M. Rahman. *Pavement crack detection using the Gabor filter*. In 2013 16th International IEEE Conference on Intelligent Transportation Systems - (ITSC), pages 2039–2044, October 2013.

[Schölkopf01] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola & Robert C. Williamson. *Estimating the Support of a High-Dimensional Distribution*. Neural Comput., vol. 13, no. 7, pages 1443–1471, MIT Press, Cambridge, MA, USA, July 2001.

[Sheikh06] H. R. Sheikh, Z. Wang, L. Cormack & A. C. Bovik. *LIVE Image Quality Assessment Database*, November 2006.

[Shelhamer16] Evan Shelhamer, Jonathan Long & Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. arXiv :1605.06211 [cs], May 2016. arXiv : 1605.06211.

[Sieberth16] Till Sieberth, Rene Wackrow & Jim H. Chandler. *Automatic detection of blurred images in UAV image sets*. ISPRS Journal of Photogrammetry and Remote Sensing, vol. 122, pages 1–16, December 2016.

[Sim13] K. S. Sim, Y. Y. Kho, C. P. Tso, M. E. Nia & H. Y. Ting. *A Contrast Stretching Bilateral Closing Top-Hat Otsu Threshold Technique for Crack Detection in Images*. Scanning, vol. 35, no. 2, pages 75–87, 2013.

[Smith97] Stephen M. Smith & J. Michael Brady. *SUSAN—A New Approach to Low Level Image Processing*. International Journal of Computer Vision, vol. 23, no. 1, pages 45–78, May 1997.

[Son Hyojoo12] Son Hyojoo, Kim Changmin & Kim Changwan. *Automated Color Model–Based Concrete Detection in Construction-Site Images by Using Machine Learning Algorithms*. Journal of Computing in Civil Engineering, vol. 26, no. 3, pages 421–433, May 2012.

[Szegedy15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke & Andrew Rabinovich. *Going deeper with convolutions*. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, June 2015. ISSN : 1063-6919, 1063-6919.

[Talab16] Ahmed Mahgoub Ahmed Talab, Zhangcan Huang, Fan Xi & Liu HaiMing. *Detection crack in image using Otsu method and multiple filtering in image processing techniques*. Optik, vol. 127, no. 3, pages 1030–1033, February 2016.

[Tao18] Xian Tao, Dapeng Zhang, Wenzhi Ma, Xilong Liu, De Xu, Xian Tao, Dapeng Zhang, Wenzhi Ma, Xilong Liu & De Xu. *Automatic Metallic Surface Defect Detection and Recognition with Convolutional Neural Networks*. Applied Sciences, vol. 8, no. 9, page 1575, September 2018.

[Vapnik98] Vladimir Vapnik. *Statistical Learning Theory | Statistics Special Topics | General & Introductory Statistics | Subjects | Wiley*, 1998.

[Xiao18] Zhitao Xiao, Yanyi Leng, Lei Geng & Jiangtao Xi. *Defect detection and classification of galvanized stamping parts based on fully convolution neural network*. In Ninth International Conference on Graphic and Image Processing (ICGIP 2017), volume 10615, page 106150K. International Society for Optics and Photonics, April 2018.

[Xue-jun Xu13] Xiao-ning Zhang Xue-jun Xu. *Crack detection of reinforced concrete bridge using video image*. Journal of Central South University, vol. 20, no. 9, pages 2605–2613, September 2013.

[Xue18] Yadong Xue & Yicheng Li. *A Fast Detection Method via Region-Based Fully Convolutional Neural Networks for Shield Tunnel Lining Defects*. Computer-Aided Civil and Infrastructure Engineering, vol. 33, no. 8, pages 638–654, August 2018.

[Yamaguchi09] Tomoyuki Yamaguchi & Shuji Hashimoto. *Fast crack detection method for large-size concrete surface images using percolation-based image processing*. Machine Vision and Applications, vol. 21, no. 5, pages 797–809, February 2009.

[Yang17] L. Yang, B. Li, W. Li, Z. Liu, G. Yang & J. Xiao. *A robotic system towards concrete structure spalling and crack database*. In 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 1276–1281, December 2017.

[Yang18] Xincong Yang, Heng Li, Yantao Yu, Xiaochun Luo, Ting Huang & Xu Yang. *Automatic Pixel-Level Crack Detection and Measurement Using Fully Convolutional Network*. Computer-Aided Civil and Infrastructure Engineering, vol. 0, no. 0, 2018.

[Yeum15] Chul Min Yeum & Shirley J. Dyke. *Vision-Based Automated Crack Detection for Bridge Inspection*. Computer-Aided Civil and Infrastructure Engineering, vol. 30, no. 10, pages 759–770, 2015.

[Yi16] X. Yi & M. Eramian. *LBP-Based Segmentation of Defocus Blur*. IEEE Transactions on Image Processing, vol. 25, no. 4, pages 1626–1638, April 2016.

[Yu17] H. Yu, W. Yang, H. Zhang & W. He. *A UAV-based crack inspection system for concrete bridge monitoring*. In 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pages 3305–3308, July 2017.

[Yue17] Guanghui Yue, Chunping Hou, Ke Gu & Nam Ling. *No reference image blurriness assessment with local binary patterns*. Journal of Visual Communication and Image Representation, vol. 49, pages 382–391, November 2017.

[Yunqiang Chen01] Yunqiang Chen, Xiang Sean Zhou & T. S. Huang. *One-class SVM for learning in image retrieval*. In Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205), volume 1, pages 34–37 vol.1, October 2001.

[Zhang13] Lilian Zhang & Reinhard Koch. *An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency*. Journal of Visual Communication and Image Representation, vol. 24, no. 7, pages 794–805, October 2013.

[Zhang14] Wenyu Zhang, Zhenjiang Zhang, Dapeng Qi & Yun Liu. *Automatic Crack Detection and Classification Method for Subway Tunnel Safety Monitoring*. Sensors, vol. 14, no. 10, pages 19307–19328, October 2014.

[Zhang17] Allen Zhang, Kelvin C. P. Wang, Baoxian Li, Enhui Yang, Xianxing Dai, Yi Peng, Yue Fei, Yang Liu, Joshua Q. Li & Cheng Chen. *Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network*. Computer-Aided Civil and Infrastructure Engineering, vol. 32, no. 10, pages 805–819, October 2017.

[Zhao17] Xuefeng Zhao & Shengyuan Li. *A Method of Crack Detection Based on Convolutional Neural Networks*. Structural Health Monitoring 2017, vol. 0, no. shm, 2017.

[Zhu08] Z. Zhu & I. Brilakis. *Defects Detection & Assessment in Concrete Surfaces*. In ResearchGate, pages 441–450, January 2008.

[Zou12] Qin Zou, Yu Cao, Qingquan Li, Qingzhou Mao & Song Wang. *CrackTree : Automatic crack detection from pavement images*. Pattern Recognition Letters, vol. 33, no. 3, pages 227–238, February 2012.