Inpainting de modèles 3D pour la réalité diminuée : "couper/coller" réaliste pour l'aménagement d'intérieur

Julien Fayer

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Informatique et Télécommunication

Présentée et soutenue par :

M. JULIEN FAYER

le vendredi 19 avril 2019

Titre :

Inpainting de modèles 3D pour la réalité diminuée: "couper/coller" réaliste pour l'aménagement d'intérieur

---

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (I.R.I.T.)

Directeur(s) de Thèse :

MME GÉRALDINE MORIN

M. SIMONE GASPARINI

Rapporteurs :

M. DAVID TSCHUMPERLE, CNRS

M. JEAN-MICHEL DISCHLER, UNIVERSITE STRASBOURG

Membre(s) du jury :

M. MICHEL DEVY, LAAS TOULOUSE, Président

M. BENJAMIN COUDRIN, LAAS TOULOUSE, Invité

Mme GÉRALDINE MORIN, INP TOULOUSE, Membre

M. ROMAIN VERGNE, UNIVERSITE GRENOBLE ALPES, Membre

M. SIMONE GASPARINI, INP TOULOUSE, Membre

M. STEPHANE MERCIER, SAS INNERSENSE, Membre

M. VINCENT CHARVILLAT, INP TOULOUSE, Invité

M. WILLIAM PUECH, UNIVERSITE DE MONTPELLIER, Membre

Remerciements

La thèse de doctorat est comme une pièce de théâtre : un acteur principal, des acteurs majeurs ainsi que toutes les personnes de l'ombre qui gravitent autour du premier. Si on a tendance à voir le doctorant comme l'acteur et le moteur principal du déroulement de la thèse, celle-ci n'aurait pu arriver à son terme sans l'aide des autres acteurs de cette pièce.
Pour cela, je tiens à remercier chaleureusement plusieurs personnes.
Premièrement, je tiens à remercier mes rapporteurs, David Tschumperlé et Jean-Michel Dischler, pour avoir accepté de lire mon manuscrit et de fournir des remarques pertinentes. Je remercie également les membres du jury : Romain Vergne, Michel Devy et William Puech pour avoir accepté de participer à mon jury de soutenance. Je remercie Stéphane Mercier qui a accepté de me faire confiance, tout d'abord comme stagiaire dans l'entreprise Innersense, puis comme doctorant à travers cette thèse. Je remercie également mes deux encadrants de thèse : Géraldine Morin et Simone Gasparini qui m'ont suivi durant ces trois années, avec exigence mais bienveillance. Je remercie aussi Benjamin Coudrin qui m'a encadré côté Innersense, malgré la distance qui nous séparait. Je remercie Vincent Charvillat qui m'a encouragé à me lancer dans cette aventure que l'on nomme doctorat et qui a lancé le projet REALISM, projet finançant ma thèse.
Mais derrière ces acteurs se cachent d'autres personnes qui m'ont accompagné durant ces dernières années. Je pense notamment à nos trois secrétaires favorites : SAM, Murielle et Annabelle qui m'ont accompagné durant les formalités administratives. Je tiens à remercier tous les membres, passés et présents, de l'équipe REVA pour l'ambiance propice à le recherche tout en permettant des moments de franche rigolade : les anciens Bastien, Yvain et Vincent, les camarades de thèse Damien, Jean, Mathieu, Sonia, Thomas, Thibault, Thierry et Arthur ainsi que les permanents Jean-Denis, Sylvie, Pierre et Axel. Je remercie également l'équipe d'Innersense, en particulier Maxime Daisy qui m'a aidé à intégrer mes travaux de recherche dans les briques d'Innersense.
Enfin, je tiens à remercier les personnes qui m'ont permis de m'octroyer des moments de détente. Je commence par les choristes de la chorale du Souffle des Ondes, qui à travers les répétitions (et les apéros) du mardi soir, me permettaient de relâcher la pression le temps d'un soir. Je remercie aussi Camille pour m'avoir encouragé et soutenu durant la dernière année, et surtout les derniers mois qui n'étaient pas les plus faciles. Enfin, je tiens à remercier ma famille et plus particulièrement mes parents, Chantal et Roland ainsi que mon frère Thomas qui m'ont toujours soutenu dans les choix que j'ai faits durant mes études et plus généralement, durant ma vie.

Résumé

Par opposition à la Réalité Augmentée qui consiste à ajouter des éléments virtuels à un environnement réel, la Réalité Diminuée consiste à supprimer des éléments réels d'un environnement. Le but est d'effectuer un rendu visuel d'une scène 3D où les éléments "effacés" ne sont plus présents : la difficulté consiste à créer une image de sorte que la diminution ne soit pas perceptible par l'utilisateur. Il faut donc venir compléter la scène initialement cachée par ces éléments, en effectuant une opération d'*inpainting* qui prenne en compte la géométrie de la pièce, sa texture (structurée ou non), et la luminosité ambiante de l'environnement. Par exemple, l'œil humain est sensible à la régularité d'une texture. L'un des objectifs d'Innersense, entreprise spécialisée dans l'aménagement virtuel d'intérieurs, est de développer un produit capable d'enlever des éléments présents dans une pièce d'intérieur. Une fois la suppression virtuelle des meubles existants effectuée , il sera alors possible d'ajouter des meubles virtuels dans l'espace laissé vacant.

L'objectif de cette thèse CIFRE est donc de mettre en place un scénario de réalité diminuée pouvant être exécuté sur un système mobile (tablette IOS ou Android) qui génère des images photo-réalistes de la scène diminuée. Pour cela, à partir d'un modèle géométrique de la pièce d'intérieur que l'on veut altérer, nous adaptons et améliorons des procédures d'effacement d'éléments d'une image appelées *inpainting* dans une image 2D. Ensuite, nous appliquons ces techniques dans le contexte 3D intérieur pour tenir compte de la géométrie de la scène. Enfin, nous analysons la luminosité pour augmenter le réalisme des zones complétées.

Dans cette thèse, nous rappelons d'abord les différents travaux académiques et les solutions industrielles existantes. Nous évoquons leurs avantages et leurs limites. Nous abordons ensuite les différentes techniques d'*inpainting* existantes pour introduire notre première contribution qui propose d'adapter une des méthodes de l'état de l'art pour prendre en compte de la structure du motif de la texture. La problématique de la luminosité est ensuite abordée en proposant un processus qui traite séparément la texture et la variation de la luminosité. Nous présentons ensuite une troisième contribution qui propose un critère de confiance basé sur des considérations radiométriques pour sélectionner une information selon sa qualité dans le processus d'*inpainting*. Nous proposons une dernière contribution basée sur la complétion de texture de modèles 3D non planaires reconstruits à partir de peu d'images et donc présentant une texture incomplète. Enfin, nous montrons les applications développées grâce à ces travaux dans le contexte des scènes d'intérieur considérées par Innersense.

Abstract

In contrast to Augmented Reality, which consists in adding virtual elements to a real environment, Diminished Reality consists in removing real elements from an environment. The goal is to visually render a 3D scene where the "deleted" elements are no longer present : the difficulty is to create an image so that the processing is not perceptible to the user. It is therefore necessary to complete the scene initially hidden by these elements, by performing an *inpainting* operation that takes into account the geometry of the part, its texture (structured or not), and the ambient brightness of the environment. For example, the human eye is sensitive to the regularity of a texture. One of the objectives of Innersense, a company specializing in virtual interior design, is to develop a product that can remove elements from an interior room. Once the virtual removal of existing furniture has been completed, it will then be possible to add virtual furniture in the vacant space.

The objective of this CIFRE thesis is therefore to set up a scenario of diminished reality that can be executed on a mobile system (IOS or Android tablet) that generates photorealistic images of the diminished scene. To do this, based on a geometric model of the interior part that we want to alter, we adapt and improve procedures for erasing elements of an image called *inpainting* in a 2D image. Then, we apply these techniques in the 3D indoor context to take into account the geometry of the scene. Finally, we analyze the brightness to increase the realism of the completed areas.

In this thesis, we first review the various academic works and existing industrial solutions. We discuss their advantages and limitations. We then discuss the different existing *inpainting* techniques to introduce our first contribution which proposes to adapt one of the state of the art methods to take into account the structure of the texture pattern. The problem of brightness is then discussed by proposing a process that deals separately with texture and variation of brightness. We then present a third contribution that proposes a confidence criterion based on radiometric considerations to select information according to its quality in the *inpainting* process. We propose a last contribution based on the texture completion of non-planar 3D models reconstructed from few images and therefore presenting an incomplete texture. Finally, we show the applications developed through this work in the context of the interior scenes considered by Innersense.

Table des matières

Première partie

Contexte et veille

Chapitre 1

La Réalité Diminuée : contexte et enjeux

Sommaire

Dans ce chapitre, nous allons introduire le contexte de la thèse en détaillant les enjeux économiques de l'entreprise Innersense. Ensuite, nous rappelons la notion de Réalité Diminuée à travers le prisme de la Réalité Augmentée. Nous présentons ensuite un état de l'art académique et industriel des concepts et des produits voisins du sujet de la thèse. Nous faisons ressortir les limites de ces concepts et produits voisins. Nous essayons enfin de répondre à ces limites à travers les contributions décrites dans les chapitres suivants.

1.1 Contexte de la thèse

Les travaux présentés dans cette thèse s'inscrivent dans le cadre d'une thèse CIFRE issue d'une collaboration entre l'équipe de recherche REVA du laboratoire IRIT et la société toulousaine Innersense.
Innersense est une jeune entreprise créée en 2014, fondée sur le constat suivant : pourquoi aujourd'hui, avec toute la technologie disponible, réaménager son intérieur constitue encore une tâche aussi difficile ? Ainsi a germé l'idée fondatrice de la société : faire des technologies mobiles le nouvel outil standard pour l'aménagement des espaces, permettant aux utilisateurs de se passer de la phase de mesure habituelle, de concevoir leur projet et de se projeter dans leurs futurs achats. La 3D et la Réalité Augmentée constituent les fondements de ce nouvel usage, permettant aux utilisateurs d'interagir avec les produits de l'aménagement ainsi que de les visualiser dans leur pièce existante. La particularité des applications Innersense repose ainsi sur le fait de pouvoir réaliser des aménagements in situ, et non pas dans une pièce virtuelle comme c'est le cas dans tous

les logiciels d'aménagements traditionnels. La qualité du rendu graphique, l'interaction entre l'aménagement virtuel et la pièce réelle de l'utilisateur, ainsi que l'ergonomie naturelle et intuitive des outils réalisés constituent des priorités stratégiques pour Innersense. La Réalité Diminuée constitue une étape supplémentaire par rapport au concept de Réalité Augmentée. Il ne s'agit plus uniquement de superposer des objets virtuels sur un environnement réel, mais d'altérer une capture de cet environnement. Dans une problématique d'aménagement d'espaces, où les environnements réels des utilisateurs sont souvent encombrés ou réduits, difficilement modifiables, les usages peuvent être nombreux. En ce sens, les problématiques de préparation des données, de segmentation des objets 3D pour leur effacement, et de reconstruction réaliste des parties effacées seront au cœur du sujet.

1.1.1 Enjeux économiques

Le secteur de l'ameublement représente à lui seul un marché considérable : 9,27 milliards en France en 2013, dont 3 milliards pour l'ameublement professionnel. Par ailleurs, l'aménagement d'espaces est un domaine encore bien plus large, englobant les secteurs du bricolage, les aménagements de la maison (piscine, vérandas, etc.) et l'aménagement professionnel. La base d'utilisateurs à toucher est ainsi très large. Innersense développe des outils pour les professionnels du secteur, de toutes tailles, en leur permettant d'améliorer leur relation avec le client final par des outils de pointe. Les technologies travaillées ici sont inédites dans le secteur des outils d'aménagement d'espace et permettront d'avoir un impact considérable sur le secteur, permettant de réinventer les outils d'aménagement utilisés au jour le jour par de nombreux professionnels ainsi que tous leurs clients. L'engagement sur ces thématiques de recherche (segmentation automatique de la pièce, des objets la composant, puis leur effacement) constitue une vraie priorité stratégique pour Innersense, afin de se distinguer de la concurrence dans le temps, renforcer l'utilité concrète de ses produits auprès du grand public et asseoir sa position sur le marché émergent de l'essayage virtuel pour l'aménagement des espaces.

1.1.2 La R&D à Innersense

Innersense mène depuis ses débuts une stratégie « active » en matière de R&D, réalisée bien sûr en proportion avec ses moyens de jeune entreprise. Une première thèse CIFRE, lancée en mars 2015 avec le LAAS/ CNRS et finie fin 2018, a porté sur les technologies de localisation et reconstruction (SLAM) à partir d'images caméra disponibles sur smartphone et sur les futurs capteurs RGB-D. Il s'agit d'une brique technologique rattachée à la « perception » de l'environnement utilisateur, permettant d'extraire de l'information 3D sur celui-ci.

Par ailleurs, Innersense est lauréate depuis fin 2015 d'un appel à projet R&D de la Région Midi-Pyrénées, réalisé en consortium avec les équipes du LAAS/CNRS et de l'IRIT/CNRS, intitulé REALISM et portant sur la thématique de la « Réalité Diminuée ». Le sujet de cette thèse s'inscrit ainsi en cohérence complète avec les démarches déjà réalisées et en cours, en contribuant à l'étape qui fait suite à la perception de l'environnement, qui consiste en l'exploitation et l'édition de ces données 3D à des fins d'aménagement. Cette partie d'exploitation des données est ainsi explicitement distinguée de la partie d'acquisition, afin de pouvoir traiter indifféremment des données en provenance de la brique SLAM / reconstruction 3D, mais également les données obtenues par les nouveaux capteurs RGB-D arrivant sur le marché. De cette manière, les résultats seront exploitables sur l'ensemble des terminaux mobiles du marché pour les années à venir, quoiqu'il arrive.

1.1.3 La problématique de la thèse

L'objectif d'Innersense à travers cette thèse est d'améliorer l'aménagement d'intérieur, limité pour le moment à l'ajout de meubles, en proposant la fonction d'édition des meubles qui sont déjà présents dans l'environnement. Cela passe donc par la fonction suppression, c'est-à-dire pouvoir effacer des meubles de la pièces afin de les remplacer par les meubles virtuels affichés via la Réalité Augmentée. Les contraintes et marges de manœuvre techniques sont assez fortes, délimités par des cas d'usage bien identifiés :

- les algorithmes utilisés doivent fonctionner sur une plateforme mobile, iOS ou Android, aux capacités de calcul limitées, tout en fournissant des résultats dans des délais acceptables par un utilisateur (temps interactif) ;
- l'utilisateur est grand public, novice ; l'interaction avec celui-ci doit donc être la plus simple possible. Cependant, cette interaction avec l'utilisateur existe bel et bien, et il peut être pertinent de faire appel à lui si nécessaire ;
- les technologies utilisées doivent être fiables dans les conditions d'usage, sans nécessiter d'expertise particulière, en particulier pour la prise d'images d'intérieur avec la caméra intégrée au smartphone / tablette ;
- pour que l'expérience utilisateur soit satisfaisante, le réalisme visuel du résultat doit constituer un critère d'évaluation important ;
- on distingue deux phases d'usage par l'utilisateur : une phase « en ligne », où l'utilisateur utilise sa tablette pour faire de l'essayage virtuel en temps réel sur son intérieur, et une phase de travail, sur laquelle l'utilisateur vient concevoir et réaliser son aménagement en travaillant sur scène statique, à partir de captures de données réalisées précédemment.

La deuxième phase fait partie intégrante du sujet de thèse vu qu'il faut pouvoir effectuer le traitement qui **efface** les objets que l'utilisateur veut enlever de sa pièce et qui **génère** un rendu **photoréaliste** de la scène sans les objets "enlevés". La problématique de la thèse est donc d'améliorer les méthodes de complétion dans l'image, ou d'*inpainting*, pour qu'elles puissent gérer la **structure** de la texture, la **luminosité** de la pièce et la **résolution** du rendu.

1.2 La Réalité Diminuée

Dans cette section nous allons décrire le contexte de la Réalité Diminuée en la situant par rapport aux autres Réalités étudiées en vision par ordinateur. Nous allons mettre en évidence les défis à relever pour effectuer un scénario de Réalité Diminuée ainsi que les approches qui essaient de résoudre les problématiques associées. Enfin, nous parlons des solutions industrielles existantes qui se rapprochent d'une application de Réalité Diminuée.

1.2.1 Classification des différentes « Réalités »

Avant de continuer, rappelons les différentes réalités utilisées en vision par ordinateur en commençant par la Réalité Augmentée.
Tout d'abord, La Réalité Augmentée est à la jonction entre la télé-présence (la "vraie" Réalité ou filtrée à travers une caméra ou un autre dispositif) et la Réalité Virtuelle.

Définition 1.2.1. La *Réalité Virtuelle* simule la présence physique d'un utilisateur dans un environnement artificiellement généré par des logiciels. Elle crée donc un environnement 3D avec lequel l'utilisateur peut interagir (voir figure 1.2).

FIGURE 1.1 – L'application AR de Nintendo sur 3DS (crédit de l'image : Nintendo).

En Réalité Augmentée, l'utilisateur peut voir le monde réel, avec des objets virtuels superposés par-dessus le monde réel. La Réalité Augmentée, au lieu de remplacer la réalité, l'enrichit. Du point de vue de l'utilisateur, les objets virtuels et réels cohabitent dans le même espace.
Une première définition concernant la Réalité Augmentée a été proposée par Ronald T. Azuma [Azu97].

Définition 1.2.2. La *Réalité Augmentée* (abrégée RA) est un système ayant les trois caractéristiques suivantes :

1. Il combine le réel et le virtuel ; au monde réel en trois dimensions doivent être intégrés des entités également en trois dimensions
2. Il est interactif en temps réel.
3. Il est recalé dans un espace à trois dimensions ; cela permet de faire coïncider visuellement les entités virtuelles avec la réalité.

Donnons quelques exemples d'applications et notons lesquels satisfont la définition donnée par Azuma.

Exemple 1.2.1. La console 3DS de Nintendo a une application qui utilise des marqueurs représentant des personnages de l'univers de Nintendo pour les visualiser dans le monde réel (voir figure 1.1). Il est possible de les déplacer et de les faire interagir entre eux. Cette application est une application de RA car elle remplit les trois conditions :

1. elle combine les personnages virtuels avec les objets déjà présents dans le monde réel ;
2. elle fait interagir ces personnages en temps-réel ;
3. le marqueur RA posé sur une surface horizontal permet de définir un espace 3D pour permettre le déplacement des personnages.

Exemple 1.2.2. L'application *Pokémon Go* qui permet de capturer des Pokémon (*cf.* figure 1.3) dans le monde réel n'est pas une application de RA. Bien qu'elle remplisse les conditions (1) et (2) de la définition 1.2.2, elle n'a pas recalée dans un repère 3D : les Pokémons semblent flotter dans les airs et sont toujours face à la caméra même lorsque l'on tourne autour.

À ce stade, nous avons trois types de situations : la télé-présence (tout est réel), l'environnement virtuel et la RA. On peut noter que tous les trois remplissent les conditions (2) et (3) qui sont discriminantes (soit les situations les valident, soit elles ne les valident

FIGURE 1.2 – Exemple de rendu d'une application de réalité virtuelle via un casque de réalité virtuelle.

pas). La première cependant peut être variable selon la définition que l'on donne aux mots " réel " ou " virtuel ". En effet, on peut considérer la tasse et le bloc-note affichés sur l'écran de la 3DS de la figure 1.1 comme virtuels (car affichés sur l'écran de la console) alors qu'ils sont effectivement présents dans la réalité. À l'inverse, un meuble généré à partir d'un modèle 3D et de textures peut avoir un rendu photo-réaliste tel qu'il peut être assimilé à un objet réel. Pour clarifier cela, MILGRAM et KISHINO [MK94] proposent une définition claire de ce qui est réel ou virtuel :

Définition 1.2.3. Un objet réel est un objet qui a une existence objective et réelle. Un objet virtuel est un objet qui existe en essence, mais pas de manière objective ou réel.

MILGRAM et KISHINO proposent d'ailleurs d'étendre la définition d'Azuma via une taxonomie (voir figure 1.4) qui définit un continuum entre le réel, situé à gauche de l'axe et le virtuel, situé à droite de l'axe. Les trois situations sont donc présents sur cet axe : la télé-présence à l'extrême-gauche, l'environnement virtuel à l'extrême-droite et la Réalité Augmentée au centre-gauche. Cet axe permet de définir une nouvelle réalité situé au centre-droite : la Virtualité Augmentée.

Définition 1.2.4. La *Virtualité Augmentée* est une application définie et recalée dans un espace à trois dimensions, interactif en temps réel et qui combine des objets réels dans un environnement virtuel.

On peut également affiner la définition de la Réalité Augmentée par rapport à cette taxinomie.

Définition 1.2.5. La Réalité Augmentée est une application définie et recalée dans un espace à trois dimensions, interactif en temps réel et qui combine des objets virtuels dans un environnement réel.

L'ensemble des applications qui valident la définition 1.2.2 sont appelés dorénavant des applications de Réalité Mixte.

Cependant, cela ne suffit pas à modéliser l'ensemble des possibilités (comment caractériser l'altération ou la suppression d'un objet réel par exemple ?).

FIGURE 1.3 – Exemple de rendu de *Pokémon Go* (crédit image : Pokémon Company).

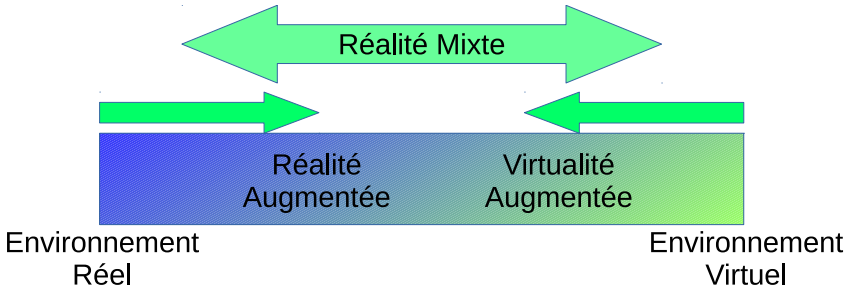


FIGURE 1.4 – Les différentes réalités selon la taxonomie de Milgram. Plus on se rapproche de la gauche de l'axe, plus on est proche de la télé-présence. Et plus on se rapproche de la droite de l'axe, plus on est proche d'un environnement virtuel.

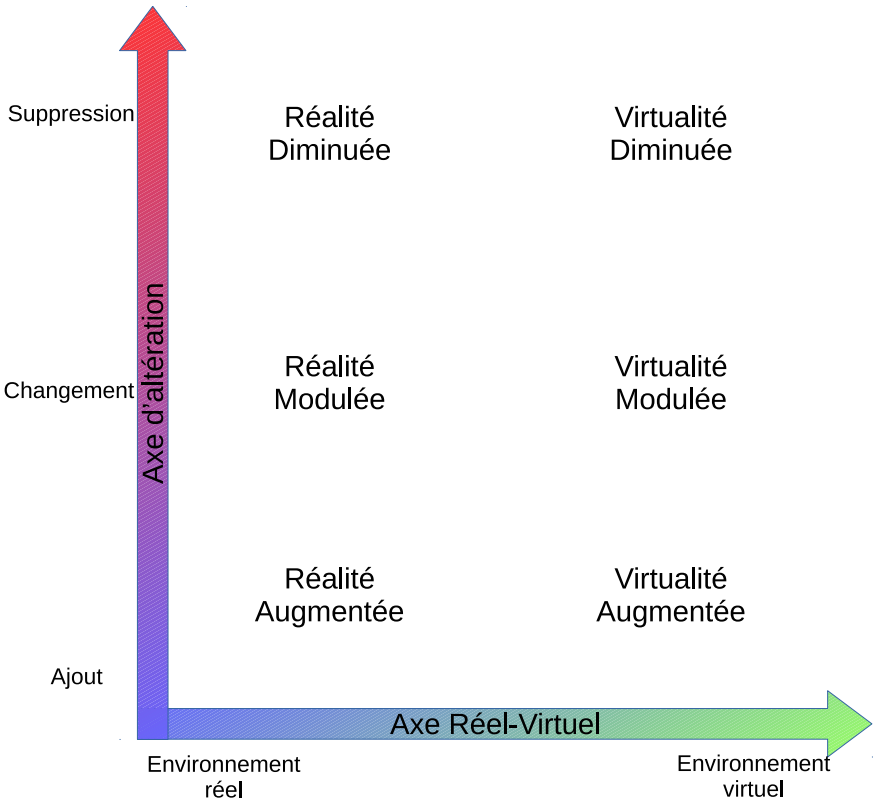


FIGURE 1.5 – Les différentes réalités selon deux axes : axe Réel-Virtuel en abscisse et axe d'altération en ordonnée. En haut de l'axe d'altération se trouvent les systèmes diminuant l'environnement (suppression des objets réels). Au milieu de l'axe se trouvent les systèmes altérant l'environnement (en changeant la texture d'un objet réel ou en le déplaçant dans l'espace 3D). En bas de l'axe se trouvent les systèmes augmentant l'environnement (ajout d'objets virtuels).

En plus de l'axe Réel-Virtuel, MANN [Man02] propose un nouvel axe dit d'altération qui rajoute trois degrés d'altération : l'augmentation ou l'ajout, l'altération et la diminution.

La figure 1.5 nous montre les différentes situations selon le type d'environnement et le type d'altération appliquée aux objets.

Nous définissons le système qui concerne cette thèse, à savoir la Réalité Diminuée.

Définition 1.2.6. La *Réalité Diminuée* (abrégée RD) est un système de Réalité Mixte (interactif temps-réel ; défini et recalé dans un espace 3D) qui consiste à supprimer des éléments réels d'un environnement réel.

Maintenant que nous avons défini les différentes Réalités, nous abordons dans la prochaine section les défis à remplir en vision par ordinateur concernant la RD.

1.2.2 Les défis de la Réalité Diminuée

Dans cette sous-section, nous parlons des défis à relever que demandent la Réalité Diminuée. Nous évoquons d'abord la problématique de l'effacement d'objet. Ensuite, nous parlons de l'intégration des traitements d'effacement dans le modèle 3D. Nous évoquons l'apport des capteurs qui peuvent récupérer de l'information 3D et les raisons

FIGURE 1.6 – Exemple de rendu d'une application de réalité augmentée (crédit image : Microsoft).

FIGURE 1.7 – Exemple de rendu de *SwapUp* : une application de réalité diminuée d'Innersense présentée dans le chapitre 8.

de leur abandon en début de thèse. Enfin, nous mentionnons les autres problèmes n'ayant pas été abordés dans cette thèse.

Effacement de l'objet : la problématique de la complétion de la zone occultée

La première problématique, probablement la plus immédiate concerne l'effacement d'un objet. Plusieurs méthodes ont été proposés pour combler cette partie laissée vide par l'effacement de l'objet. Les première techniques utilisent surtout plusieurs caméras : l'intérêt est de pouvoir connaître l'arrière-plan de l'objet que l'on veut enlever.

Zokai *et al.* [Zok+03] proposent par exemple l'utilisation de différents points de vue afin de reconstruire un arrière-plan d'un espace. Cependant, cette méthode réclame que l'objet soit suffisamment éloigné de son arrière-plan ; pour cela ils proposent de diviser la région en plusieurs zones d'égales profondeurs. Les résultats sont assez partagés : d'un côté, leur méthode fonctionne bien sur des arrières-plans plats (parallèles et non parallèles à l'image à compléter) et non-plats ; mais d'un autre côté, l'objet n'est pas enlevé en entier vu qu'il existe des portions toujours occultées quelque soit sa position (comme par exemple, le dessous de l'objet). De plus, cette technique nécessite soit plusieurs caméras, soit une seule mais on doit abandonner la possibilité du temps-réel.

Jarusirisawad *et al.* [JHS10] proposent une méthode basée également sur plusieurs caméras. Ils utilisent pour cela l'algorithme du *plane-sweep* [SH76] dans un espace projectif en considérant deux caméras calibrées comme une base de l'espace. Ils projettent un point de l'espace sur une caméra non calibrée quelconque en utilisant un tenseur trifocal (tenseur de taille $3 \times 3 \times 3$) qui établit une relation géométrique entre les trois images. Leur approche se décompose en trois étapes : définition de la position d'une caméra virtuelle, définition des plans dans l'espace projectif (dans un angle de vision d'une des caméras de la base), calcul des nouvelles images et suppression de l'objet indésirable. L'intérêt de cette méthode est de pouvoir trouver la couleur située "derrière" l'objet via l'algorithme du *plane-sweep*. Les résultats nous montrent que d'un côté, l'objet occulté est bien retiré tout en respectant la cohérence spatiale mais d'un autre côté, le rendu est parfois loin d'être satisfaisant, non seulement dans la périphérie de l'objet indésirable mais aussi dans des zones de l'espace qui ne sont jamais occultées par l'objet. De plus, comme l'approche précédente, celle-ci demande l'utilisation de plusieurs caméras, ce qui est coûteux pour une application temps réel.

Li *et al.* [Li+13] proposent une approche différente qui s'éloigne des deux précédents travaux. En effet, l'idée est d'utiliser un grand ensemble de photos disponibles sur Internet (Flickr *etc.*) prises par exemple par des touristes dont la localisation se fait par GPS (voir figure 1.8). Le GPS peut, en effet, aider à récupérer l'échelle réelle du modèle. La reconstruction, quant à elle, se fait par SfM (*Structure-from-Motion* [SSS06]). Pour texturer les objets occultés dans l'image courante, ils mappent les photos, qui ont été prises à la même position que l'image courante et où les objets sont visibles. Le mappage est effectué via une Transformation Linéaire Directe (*Direct Linear Transformation* (DLT)) [HZ04]. Le reste de l'approche reste classique avec une détection-suivi de l'objet occultant puis une suppression de l'objet en utilisant les informations du set de photos et des trames obtenues précédemment. L'avantage de cette méthode par rapport aux précédentes est la possibilité d'exécution en temps réel (il suffirait de stocker l'ensemble des photos et les informations de localisation). Cependant, l'appareil devra, pour cela, disposer de suffisamment de mémoire pour exécuter l'application. L'inconvénient majeur de cette méthode est la limitation de l'environnement : dans un environnement d'intérieur, nous devons avoir suffisamment d'images de la zone cachée par l'objet à effacer ce qui est peu réaliste du fait que les objets sont souvent adossés à d'autres.

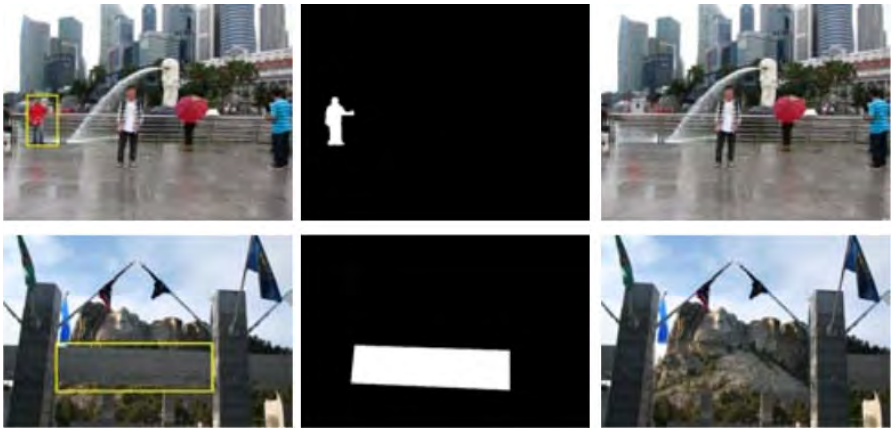


FIGURE 1.8 – Scénario de Réalité Diminuée utilisant un ensemble de photos (figure extraite de [Li+13])

Enfin, TAKEDA et SAKAMOTO [TS10] proposent de supprimer des éléments occultant un paysage . Pour cela, ils recherchent des points d'intérêt sur plusieurs images via notamment l'algorithme SIFT [Low+99]. Ils effectuent ensuite une homographie liant les deux images afin de trouver la couleur de la partie du paysage située derrière l'objet occultant. Les résultats sont satisfaisants mais cette méthode est limitée par son environnement dans la mesure où elle requiert la connaissance de la zone cachée. Si l'objet est collé à un mur, cette méthode ne pourra pas être utilisée dans notre contexte d'intérieur (l'information ne sera jamais disponible). De plus, nous avons besoin que la zone cachée par rapport à l'objet occultant soit située suffisamment loin de la caméra pour que l'homographie rende des résultats satisfaisants. Cette méthode reste donc limitée au cas de paysages (fixes et lointains) ou au cas strictement planaire.

Nous évoquons enfin de manière succincte [1] le principe de l'*inpainting*. L'*inpainting* est une technique de traitement de l'image. Elle consiste à compléter une zone dans une image à partir d'informations situées ailleurs dans l'image. Les méthodes d'*inpainting* sont très efficaces pour effacer des éléments d'une photo par exemple comme le montre la figure 1.9. Cependant, appliquée seule, l'*inpainting* trouve ses limites dans le contexte de la Réalité Diminuée. Elle aura du mal à respecter la segmentation entre les murs d'une pièce ou la déformation de la texture due à la perspective (voir figure 1.10).

Elle doit être couplée à des informations 3D si on veut pouvoir réaliser une application de Réalité Diminuée. Nous voyons les possibilités d'intégration dans la prochaine sous-section.

De manière générale, toutes ces approches, bien que proposant des techniques pour remplir la zone effacée, ne permettent pas de mettre en place une application de Réalité Diminuée (notamment sur le plan de l'interaction avec l'utilisateur ainsi que l'intégration dans un modèle 3D). Nous évoquons dans la prochaine sous-section les approches qui essaient de répondre à cette problématique.

Intégration au modèle 3D

Maintenant que nous avons définit la problématique de l'effacement, il faut discuter de la manière de l'intégrer dans un modèle 3D. En effet, nous ne voulons pas seulement effacer un objet et compléter la zone laissée vide dans une image. Nous voulons que cet

1. Le fonctionnement ainsi que les méthodes classiques de la littérature sont développées dans le chapitre 2.

FIGURE 1.9 – Effacement d'un oiseau dans une photo par une méthode d'*inpainting* [CPT03]

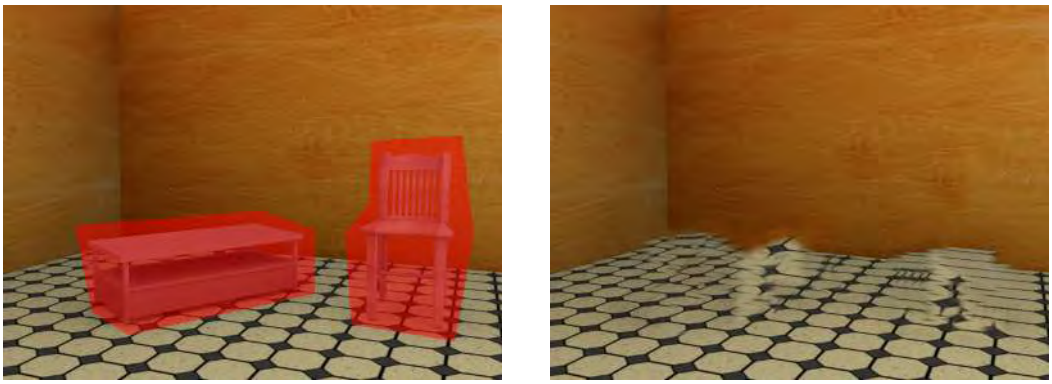


FIGURE 1.10 – Limites d'une approche d'*inpainting* seule. Gauche : photo de la scène (zone à effacer coloriée en rouge). Droite : photo après effacement des meubles par la méthode d'*inpainting* de DAISY *et al.* [DTL13].

effacement soit effectif dans tout le modèle 3D,c'est à dire, quelque soit le point de vue. Par contre, si l'utilisateur change de position, il faut que le rendu de la zone effacée soit cohérent entre la position initiale et la nouvelle position. Il est donc nécessaire d'intégrer l'effacement dans le modèle 3D. Les approches suivantes montrent les possibilités d'intégration de l'effacement.

Dans le contexte d'une application de réalité augmentée qui utilise des marqueurs pour repérer la camera dans la scène, Kawai *et al.* [Kaw+13] proposent premièrement d'effectuer de l'*inpainting* sur l'image pour générer la texture destinée à remplacer les zones cachées par les marqueurs. Une fois la texture générée, ils séparent la composante diffuse de la composante spéculaire de celle-ci et cachent le marqueur. L'apport majeur de cette approche consiste à corriger la perspective en utilisant les propriétés du marqueur (carré). On peut disposer ainsi d'un motif unifié en terme de taille et de forme. De plus, on peut déterminer facilement les correspondances entre les pixels de chaque image pour détecter les changements de luminosité. Les résultats montrent que l'utilisation de la correction de la perspective améliore le résultat final, malgré les variations de luminosité pouvant intervenir. Cependant, cette méthode trouve ses limites dans le cas où le marqueur superpose deux motifs différents. De plus, l'élément à effacer (le marqueur) n'est pas un objet 3D.

Ils étendent ensuite leur approche [KSY15] à un scénario d'effacement d'objets, désormais 3D, en incorporant des informations 3D supplémentaires. Ils effectuent un SLAM (*Simultaneous Localization And Mapping*) afin de récupérer les matrices de pose des caméra de chaque point de vue et recherchent ensuite des points d'intérêts afin d'estimer les plans géométriques présents dans la scène. Un premier intérêt des plans géométriques est de pouvoir effectuer une segmentation pour délimiter les zones de l'images selon leur appartenance à un plan. Un autre intérêt est de pouvoir calculer une homographie qui transpose l'image dans un espace où la zone du plan dans l'image est vue de manière fronto-parallèle ; on parle alors de rectification. On peut ensuite effectuer un procédé d'*inpainting* [KY12] et re-projeter le résultat dans l'espace de départ. La figure 1.11 nous montre un résultat de l'approche tout à fait convenable pour un scénario de base, où un petit objet 3D accolé aux murs est effacé.

Figure 1.11 – Résultats de l'approche de Kawai *et al.* [KSY15].

Siltanen [Sil15] propose également de diminuer la scène en utilisant des informations 3D. Le scénario diffère lors de l'acquisition de données 3D vu qu'ils utilisent un marqueur pour récupérer l'équation du plan du sol (voir figure 1.12). L'utilisateur est invité à tracer ensuite les lignes délimitant le sol des murs de la pièce ce qui permet d'obtenir les équations des plans de ces derniers. Une segmentation et une rectification sont ensuite effectuées. Par ailleurs, la variation de luminosité entre les différents pixels de l'image

rectifiée est gérée via une normalisation d'intensité. Nous parlerons du traitement de la luminosité dans le chapitre 5. Un procédé d'*inpainting* est ensuite effectué sur l'image rectifiée de chaque plan. Ils réadaptent ensuite le résultat à la luminosité ambiante [KAS10] qui est ensuite projeté dans le plan-image de la caméra.

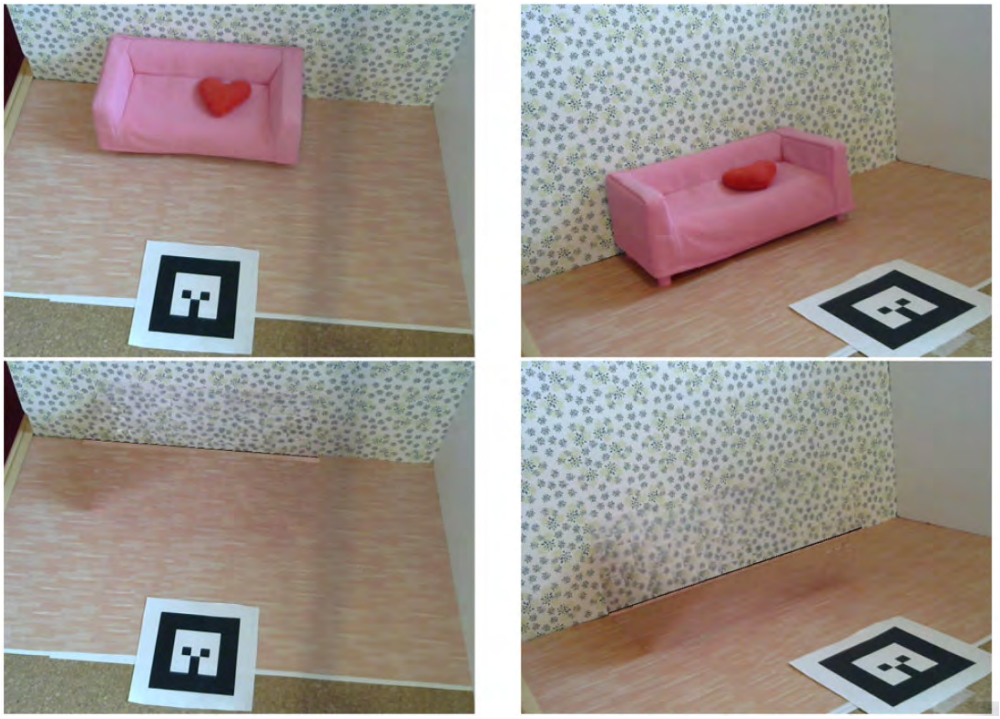


FIGURE 1.12 – Effacement d'un meuble en tenant compte de la géométrie et de la luminosité (figure extraite de [Sil15])

HERLING et BROLL [HB14] proposent une application appelée *PixMix*. *PixMix* est une méthode capable d'effectuer en temps réel par *inpainting* la suppression d'un objet suivi dans une zone d'une image après avoir été choisi par l'utilisateur (voir figure 1.14). Leur processus d'effacement est également une rectification via une homographie suivi d'un procédé d'*inpainting*. Ils ont également ajouté des fonctions de coût qui assurent la cohérence spatiale et visuelle. De plus, l'*inpainting* est couplé à un raffinement et une propagation itératifs qui consistent à appliquer une approche orientée multi-résolution : l'image est divisée par 2^n où n est un degré de profondeur. L'*inpainting* est appliqué à cette image de faible résolution puis est ré-agrandie grâce aux pyramides de Lagrange [Rad12]. Bien que les résultats soient très intéressants, ils restent limités dans la mesure où ce sont des éléments n'ayant pas de formes 3D.

Globalement, les approches évoquées ci-dessus permettent de résumer certains points. Premièrement, la géométrie de l'espace 3D est essentielle. Elle permet de localiser dans l'espace la zone à effacer, une fois celle-ci créée, et de mettre à jour les pixels représentant cette zone dans l'image ou le flux vidéo par l'information complétée via l'*inpainting*. Deuxièmement, le procédé d'*inpainting* n'est pas effectué directement dans l'image de la caméra. Il est effectué dans une image où les distorsions dues à la perspective ont été éliminées. Troisièmement, la luminosité ayant un impact pour l'utilisateur, celle-ci doit être prise en compte durant la diminution.

Cependant, la géométrie de ces approches est très limitée vu que seule une partie de la scène d'intérieur est modélisée, au lieu d'avoir une modélisation complète. Pour cela, utiliser des capteurs 3D peut permettre de fournir une géométrie plus complète de la

FIGURE 1.13 – Effacement de marqueurs en tenant compte de la luminosité ambiante (figure extraite de KAWAI *et al.* [Kaw+13]).

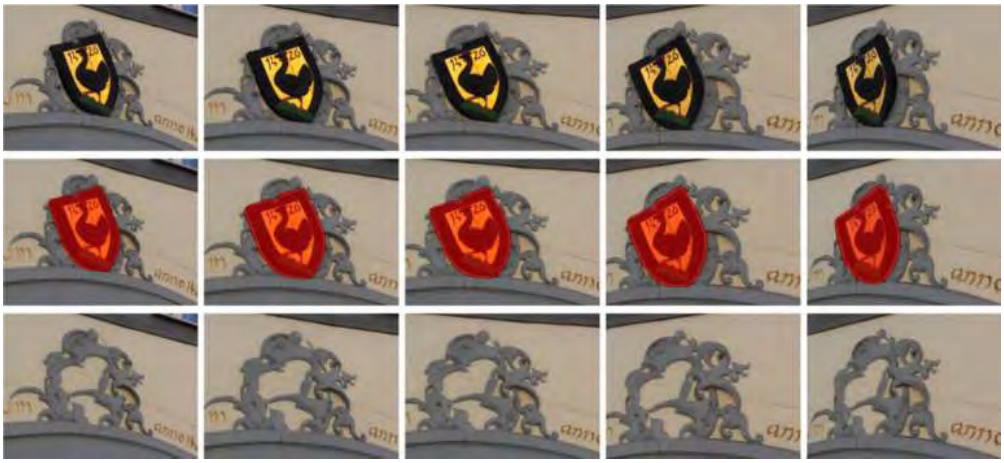


FIGURE 1.14 – *inpainting* vidéo temps réel avec *PixMix* (figure extraite de HERLING et BROLL [HB14]).

scène.

Les capteurs 3D : une révolution avortée

Il y eu depuis une dizaine d'année l'émergence d'un nouveau type de capteur : les capteurs de profondeur. Ces capteurs permettent de récupérer une carte appelée carte de profondeur qui possède pour valeur la distance entre la caméra et un point de l'environnement. Ces capteurs couplés à un capteur de couleur sont appelés capteurs RGB-D (*Red Green Blue Depth*). Ces capteurs RGB-D comme le PiCam [Ven+13], le capteur Structure [Str] ou encore la tablette Tango de Google, en récupérant la profondeur, ouvrent la porte à différentes applications de la vision par ordinateur jusqu'alors difficiles à mettre en œuvre. La possibilité de capter la profondeur ouvre la voie à de nouvelles approches notamment en terme de Réalité Diminuée.

Nous pouvons penser à la modélisation d'intérieurs qui a toujours été un défi pour la vision par ordinateur par rapport aux contraintes posées par l'environnement (luminosité limitée, manque de caractéristiques discriminantes, structures, répétitives). Nous évoquons les principales approches qui se servent des capteurs 3D pour effectuer un scénario de Réalité Diminuée.

Zhang *et al.* [ZCC16] proposent un processus basé sur un *scan* 3D d'une pièce (*cf.* figure 1.15). Leur système permet un rendu réaliste non seulement de la pièce vide dans les conditions d'éclairage d'origine, mais aussi en autorisant l'ajout de meubles, la modification des propriétés matérielles des murs ainsi que le ré-éclairage. Bien que leur rendu soit très réaliste, l'inconvénient majeur est la nécessité d'utiliser une tablette qui possède un capteur de profondeur (Tango) et d'effectuer un *scan* complet de la pièce (environ 20000 *frames*). Leur approche est donc difficilement transposable sur des systèmes mobiles qui ne possèdent qu'un capteur colorimétrique.

Technicolor [QFR18] propose une application de Réalité Diminuée basée également sur Tango (*cf.* figure 1.16). Leur approche est basée sur la pré-observation. La scène 3D est d'abord vidée réellement d'objets indésirables pour être ensuite numérisée. Elle est reconstruite sous la forme d'un modèle 3D texturé à haute résolution. Lors de l'exécution, les objets ajoutés dans une région d'intérêt sont supprimés en projetant l'arrière-plan précédemment capté. Les différences de conditions d'éclairage entre la durée de balayage et la durée d'exécution sont compensées pour obtenir des résultats homogènes. L'approche proposée n'exige aucune segmentation ou saisie manuelle autre que la définition de la région 3D d'intérêt à diminuer, et ne repose sur aucune hypothèse particulière concernant la géométrie de l'arrière-plan. L'inconvénient majeur est d'avoir besoin de l'information cachée au préalable (ce qui limite fortement son utilisation pour une scène contenant des objets difficilement déplaçables comme ceux d'une cuisine intégrée).

Nous avons aussi essayé au début de la thèse d'effectuer un déplacement d'un objet réel de la scène en lui faisant subir une translation et une rotation. L'idée est de détecter les points 3D appartenant à l'objet, de créer un maillage entre ces points puis d'effectuer la transformation. Si l'objet est scanné partiellement par le capteur de profondeur, ses parties non connues peuvent être complétées a posteriori [Li+16]. La figure 1.17 montre une preuve de concept où une boîte réelle est déplacée d'un point à un autre.

Cependant, la "révolution" de ces capteurs trouve vite ces limites. En effet, la qualité des images obtenues par des capteurs RGBD est faible par rapport aux capteurs de couleur du marché. Par exemple, la Kinect de Microsoft génère des images de profondeurs avec des résolutions deux fois plus faibles que les images couleurs qui elles même ont déjà une résolution faible de (640×480), ce qui signifie que tous les pixels ne disposent pas d'une information de profondeur.

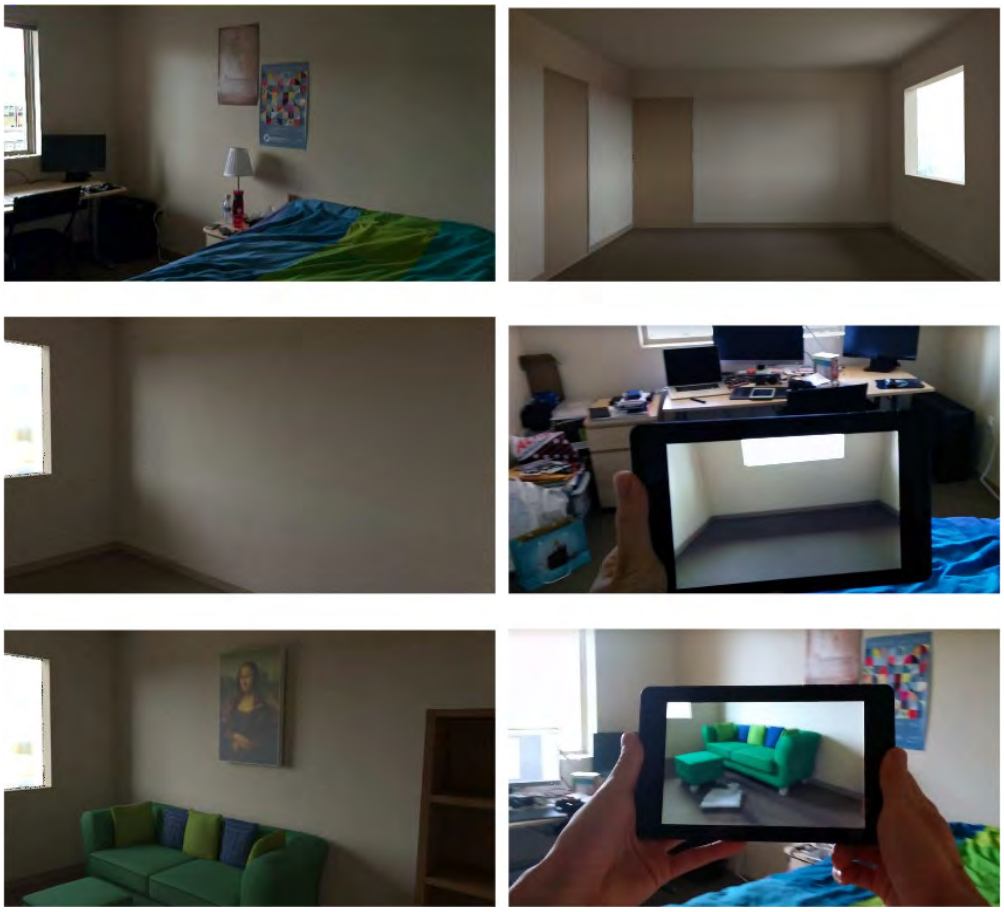


FIGURE 1.15 – Approche de Réalité Diminuée de ZHANG *et al.* (figure extraite de [ZCC16]). À partir d'un scannage RGBD d'une pièce d'intérieur, un modèle de la scène vidée est produit. Il inclut notamment la luminosité et les données des matériaux présents. Il est possible de visualiser la pièce vide depuis une pose enregistrée ou directement sur place. Leur modèle permet aussi l'édition de la scène comme l'ajout de meubles.

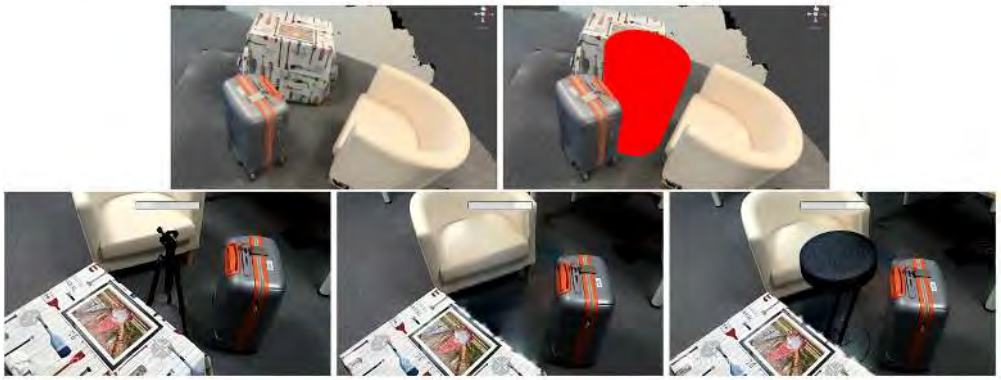


FIGURE 1.16 – Approche de Réalité Diminuée de Technicolor (figure extraite de [QFR18]). De gauche à droite, de haut en bas. Après avoir scanné une scène 3D vide et identifiée une zone d'intérêt, tout objet indésirable placé dans cette zone est supprimé. Aussi, un nouvel objet virtuel peut être inséré dans la scène au même endroit.

FIGURE 1.17 – Déplacement d'un objet à partir d'une prise RGB-D après complétion de la zone cachée par l'objet. À partir de l'image d'origine (en haut, à gauche), l'objet peut être translaté vers l'arrière (en haut, à droite). Il peut également être tourné dans un sens (en bas, à gauche) ou un autre (en bas, à droite).

Ensuite, ces capteurs ne sont pas encore prêts pour une diffusion de masse. On peut penser notamment à un coût de conception élevé ainsi qu'une autonomie très limitée. De plus, l'arrivée dernièrement des solutions de RA d'Apple (ARKit [App17]) et de Google (ARCore [Goo17]), qui peuvent fournir la 3D sans capteur supplémentaire, fait que la situation de marché n'est pas suffisamment stable pour imaginer la diffusion prochaine à grande échelle de capteurs RGBD.

Les autres problèmes non abordés durant la thèse

Bien que les problématiques les plus importantes aient évoquées ci-dessus, d'autres problèmes sont à prendre en compte. Cependant, ils ne seront pas abordés dans le cadre de cette thèse mais peuvent être considérés comme des perspectives.

Un premier point est la segmentation automatique de l'objet à effacer. Actuellement, il est nécessaire de faire intervenir l'utilisateur pour désigner l'objet à effacer en définissant manuellement un masque entourant l'objet sur un rendu de la scène. L'objectif serait de développer une méthode capable de sélectionner automatiquement un objet de la scène et de calculer automatiquement son masque dans un rendu de la scène.

Un deuxième point est la modélisation sémantique des objets réels de la scène. Cette modélisation 3D d'un objet réel permettrait de le considérer comme un objet géométrique avec un label, qui posséderait des relations avec le sol ou d'autres objets. Une telle représentation sémantique faciliterait le déplacement des objet réels. Par exemple, si nous voulons déplacer ou supprimer une table avec une lampe de bureau dessus, il serait alors possible de déplacer ou de supprimer en même temps la lampe de bureau. Notons en outre les travaux de Silberman *et al.* [Sil+12] ou Wong *et al.* [WCM15] qui essaient de résoudre cette problématique.

Un troisième point non abordé durant cette thèse est la construction d'un modèle de luminosité de la pièce comme dans l'approche de Zhang *et al.* [ZCC16]. Nous considérons la gestion de la luminosité comme un problème limité au traitement de l'image. Cependant, disposer d'un modèle complet de luminosité permet d'améliorer le rendu des zones effacées. De plus, il serait plus simple de changer la luminosité appliquée à des objets réels après déplacement ainsi que de gérer leurs ombres.

1.2.3 La Réalité Diminuée dans l'industrie

L'aménagement virtuel d'intérieur est une fonctionnalité que beaucoup de fabricants/vendeurs de meubles possèdent ou aimeraient développer.
Par exemple RHINOV [RHI] propose des solutions de rendu de visuels 3D réalistes à partir de photos d'une pièce à redécorer. Leur scénario est plus proche de la Réalité Virtuelle que de la Réalité Diminuée. Wayfair [Way], de la même manière, propose de créer un modèle 3D d'une pièce avec un rendu photo-réaliste. Mais là encore, il n'y a pas d'interaction avec un utilisateur qui serait réellement dans cet environnement.

À l'opposé, nous avons des entreprises comme Artefacto [Art] qui est spécialisée dans la Réalité Augmentée et la Réalité Virtuelle appliquées au secteur immobilier. Artefacto propose des solutions orientées vers la création de contenus 2D/3D, à la réalisation d'applications et de dispositifs de visualisation et d'interactivité.

Enfin, nous avons des entreprises qui proposent, comme Innersense, des solutions de Réalité Augmentée qui permettent d'ajouter des meubles virtuels dans un environnement réel. Nous avons par exemple Amakisa [Ami] (Réalité Virtuelle) ou IKEA [IKE] (Réalité Augmentée basée sans marqueurs) qui proposent leurs propres solutions d'aménagement virtuel.

FIGURE 1.18 – Influence de la perspective dans l'effacement d'un objet. Image de gauche : scène avec l'objet à effacer. Image de droite : scène avec l'objet effacé. Les données de basse qualité (celles situées au loin) se sont propagées dans la zone du masque.

Cependant, à l'heure actuelle, aucune entreprise ne propose un produit stable qui effectue un scénario de Réalité Diminuée capable d'effacer les meubles d'une pièce.

1.2.4 Conclusion

Ce tour d'horizon des différentes approches académiques ainsi que des produits existants a permis de mettre en évidence plusieurs points.

Premièrement, des solutions académiques avec des résultats intéressants, voire impressionnants, commencent à émerger. Or, ces solutions n'ont pas encore été transformées en brique industriel car, soit le matériel n'est pas suffisamment démocratisé pour intéresser les entreprises spécialistes du domaine de l'aménagement, soit les processus actuels ne permettent pas d'effectuer de manière simple et robuste un scénario de Réalité Diminuée qui soit utilisable par Monsieur et Madame Tout-Le-Monde.

Ensuite, si on se restreint aux solutions basées uniquement sur des capteurs colorimétriques (on exclut à partir de maintenant le cas des capteurs de profondeurs), on note à propos des applications existantes :

— Elles ne se contentent que d'effacer un seul objet, généralement de petite taille. Or, on veut éventuellement effacer tous les éléments d'une pièce (comme une cuisinière qui prend une grande partie de la place, laissant moins d'informations disponibles).
— Les méthodes d'*inpainting* utilisées (notamment par SILTANEN [Sil15]) ne prennent pas compte de la variété des textures d'intérieur. Dans l'exemple de la figure 1.12, on constate (en regardant de plus près) que la texture du mur a un motif régulier. Le résultat, bien que convenable, ne respecte pas entièrement la structure du motif de la texture.
— La luminosité peut être mal propagée, notamment dans la figure 1.12 où l'on voit que l'ombre à gauche n'a pas été effacée correctement.
— Les problèmes de résolution dans l'*inpainting* dus à la rectification peuvent générer, selon les situations, du flou à cause de la propagation des données ayant une qualité plus faible comme montré dans la figure 1.18.

1.3 Contributions principales de la thèse

La thèse propose diverses améliorations d'un scénario de Réalité Diminuée à partir d'un ensemble de prises de vue, de leurs poses caméra et d'une géométrie de la pièce considérée. Pour cela, plusieurs contributions sont détaillées dans les chapitres suivants :

— Après un état de l'art spécifique concernant l'*inpainting* développé dans le chapitre 2, nous proposons, dans le chapitre 3, une **approche *a contrario*** pour classer les textures selon la méthode de complétion la plus adaptée. Nous proposons ensuite une **nouvelle approche d'*inpainting*** qui se sert d'une initialisation contrainte par la structure de la texture pour mieux compléter la zone masquée.
— Dans le chapitre 4, nous proposons de prendre en compte la **variation de la résolution** [Fay+18e] d'une image rectifiée pour éliminer les effets de flou dus à la perspective.
— Dans le chapitre 5, nous traitons de la gestion de la luminosité à travers une formalisation essentiellement basée sur du traitement d'images. Nous améliorons la propagation de la luminosité en complétant les isocontours traversant la zone masquée.

Une fois ces contributions expliquées, nous proposons les applications suivantes :

— Dans le chapitre 6, nous détaillons **une application de réalité diminuée** dans laquelle des éléments d'une pièce rectangulaire sont effacés à partir de quelques prises de vue.
— Dans le chapitre 7, nous proposons **une extension à la géométrie non planaire** [Fay+18d] en complétant les zones de texture non visibles d'un objet 3D modélisé par des surfaces canal et reconstruit à partir de quelques prises de vue.
— Enfin dans le chapitre 8, nous détaillons *SwapUp* qui est **une application temps réel de Réalité Diminuée** développée par l'équipe R&D d'Innersense et qui intègre les contributions développées durant cette thèse.

Chapitre 2

L'*inpainting* : état de l'art et limites

Sommaire

Dans ce chapitre, nous allons aborder plus en détails les différentes méthodes de complétion. Après cela, nous mettrons en évidence les limites d'un scénario de Réalité Diminuée seulement basé sur une approche d'*inpainting*.

2.1 Introduction

L'*inpainting* (littéralement « peindre à l'intérieur ») consiste à remplacer une zone d'une image en utilisant l'information présente ailleurs dans l'image ou dans d'autres images. Les deux principaux objectifs de l'*inpainting* sont les suivants :

— la restauration d'image. Il s'agit d'enlever des défauts souvent de petite taille (rayures) et répartis plus ou moins uniformément dans l'image (bruit) comme illustrée par la figure 2.1.

— l'effacement d'éléments. Il s'agit d'enlever des zones de taille moyenne et en quantité limitée qui représentent un objet ou une personne de l'image comme illustrée par la figure 2.2.

Le procédé en soit n'est pas récent étant donné qu'il a été utilisé sur des peintures ou des photographies. Cependant, le procédé était essentiellement manuel vu qu'une personne

devait elle-même recompléter la zone indésirable. Le développement et l'universalisation de l'outil informatique permet d'exécuter rapidement et sans l'intervention d'un utilisateur une méthode d'*inpainting* sur une image donnée. Dans la suite du chapitre, nous allons

— formaliser le procédé d'*inpainting* tel qu'il est défini dans la littérature ;
— présenter les principales méthodes utilisées ;
— montrer les applications adéquates pour chacune d'entre elles.

FIGURE 2.1 – Restauration d'une vieille photographie (issue de BERTALMIO *et al.* [Ber+00])

FIGURE 2.2 – Voroshilov, Molotov, Staline et Iejov venus examiner les travaux du canal de la Volga à Moscou. Après l'élimination de Iejov en 1939, celui-ci disparaît de la photographie jusqu'à la fin de l'Union soviétique en 1991 (photos du domaine public).

2.2 Formalisation du problème

Soit I une image définie sur un domaine O de $\mathbb{P}^2$ et à valeurs dans $\mathbb{R}^n$.

$$I : O \to \mathbb{R}^n \tag{2.1}$$
$$p \mapsto I(p) \tag{2.2}$$

où :

— $\mathbb{P} = \mathbb{N}$ dans le cas d'une image discrète et $\mathbb{P} = \mathbb{R}$ dans le cas d'une image continue ;
— $n = 1$ dans le cas d'une image en niveau de gris et $n = 3$ dans le cas d'une image tri-chromatique.

Pour la suite du manuscrit, nous ne considérons que des images discrètes. On appelle pixel un élément de O. On définit pour chaque pixel $p = (x_1, y_1)$ son 4-voisinage $\mathcal{V}_p^4$ sur O :

$$\mathcal{V}_p^4 = \{q = (x_2, y_2) \in O, |x_1 - x_2| + |y_1 - y_2| = 1\}. \tag{2.3}$$

On définit également son 8-voisinage $\mathcal{V}_p^8$

$$\mathcal{V}_p^8 = \{q = (x_2, y_2) \in O, \max(|x_1 - x_2|, |y_1 - y_2|) = 1\}. \tag{2.4}$$

Nous notons $\Omega \subset O$ la zone de l'image à compléter à partir de la zone connue $O \backslash \Omega$. Nous appelons *masque* l'image suivante :

$$f_I : O \to \{0, 1\} \tag{2.5}$$
$$p \mapsto \mathbb{1}_\Omega \tag{2.6}$$

où $\mathbb{1}$ est la fonction indicatrice.

Nous notons également $\partial\Omega_i$ la frontière intérieure de Ω c'est-à-dire l'ensemble des pixels de Ω qui possède au moins un voisin de $O \backslash \Omega$. Aussi, nous notons $\partial\Omega_e$ la frontière extérieure c'est-à-dire l'ensemble des pixels de $O \backslash \Omega$ qui possède au moins un voisin de Ω. La figure 2.3 montre un exemple avec les différentes régions.

L'*inpainting* consiste à trouver une fonction f_I définie ainsi :

$$f_I : O \to \mathbb{R}^n \tag{2.7}$$
$$p \mapsto \begin{cases} I(p), & p \in O \backslash \Omega \\ f_I(p), & p \in \Omega \end{cases} \tag{2.8}$$

qui rapproche I d'un résultat qualitativement acceptable par l'œil humain (vu l'absence de mesures d'évaluation quantitative fiables [AS09]). La zone Ω est reconstruite à partir des données présentes dans la zone connue $O \backslash \Omega$ de l'image I via f_I.

Remarque 2.2.1. La fonction f_I est indicée par l'image I pour insister sur le fait que la fonction f_I va donc attribuer pour chaque pixel du masque une valeur calculée à partir des données connues uniquement.

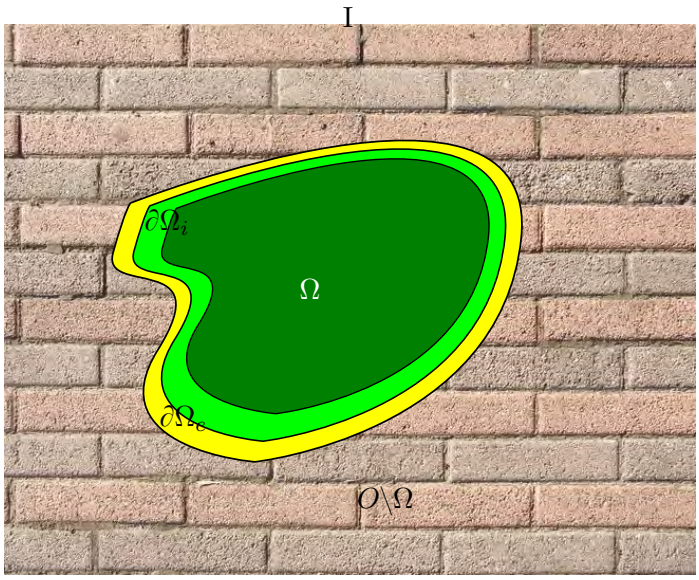


FIGURE 2.3 – Schéma représentant les différentes notations. Les zones vertes (claire et foncée) font partie du masque Ω. La zone jaune $\partial\Omega_e$ est la frontière extérieure du masque et ne fait donc pas partie du masque Ω au contraire de la frontière intérieure $\partial\Omega_i$

.

Plusieurs méthodes pour estimer la fonction f_I existent et sont regroupées en plusieurs catégories [Buy+15]. Nous présentons l'idée principale de chaque catégorie et nous énonçons les approches courantes dans la prochaine section.

2.3 Les différentes catégories d'*inpainting*

Après avoir énoncé le problème d'*inpainting*, nous présentons les approches les plus courantes classées par catégorie et selon la méthode de complétion.

2.3.1 Les méthodes basées diffusion

Les approches de cette catégorie ont été les premières à être développées [Ber+00 ; CS01a]. Les méthodes basées diffusion consistent à remplir la région manquante d'une image à travers un processus de diffusion qui propage l'information présente sur les bords du masque. On peut considérer f_I comme une fonction d'interpolation de la zone masquée Ω à partir des données connues frontalières de $\partial\Omega_e$.

Soit $I : O \to \mathbb{R}$ une image et Ω un masque à compléter. L'approche générale consiste à effectuer une itération sur n sur une image de départ $I^0 = I$ selon une fonction de diffusion f_t :

$$I^{n+1}(i,j) = I^n(i,j) + \Delta t f_t(I^n(i,j)) \tag{2.9}$$

où Δf est le taux de progrès. Par exemple, chez Bertalmio *et al.* [Ber+00], on a :

$$f_t(I^n(i,j)) = \overrightarrow{\delta L^n}(i,j) \cdot \overrightarrow{N}^n(i,j) \tag{2.10}$$

où $\overrightarrow{\delta L^n}$ est l'opérateur gradient du laplacien de I^n :

$$\overrightarrow{\delta L^n}(i,j) = (L^n(i+1,j) - L^n(i-1,j), L^n(i,j+1) - L^n(i,j-1)) \tag{2.11}$$

$$L^n(i,j) = I^n_{xx}(i,j) + I^n_{yy}(i,j) \tag{2.12}$$

et $\overrightarrow{N}$ est la direction des courbes isophotes (lignes où les pixels ont le même niveau de gris).

Chan et Shen [CS01a] proposent aussi un modèle basé sur les Équations Différentielles Partielles (EDP) où le coefficient de conductivité dépend de la courbure des isophotes. Tschumperlé et Deriche [TD05] étoffe ce modèle en proposant une diffusion basée également sur les EDP qui se fait de manière isotrope dans les zones homogènes et de manière anisotrope au niveau des contours. Bornemann et März [BM07] proposent un modèle basé sur le transport des valeurs connues de l'image dont la direction est estimée via des tenseurs de structure. Shen et Chan [SC02] proposent un modèle basé sur la Variation Totale (*Total Variation (TV)*) où une fonction résultat définie pour minimiser une fonctionnelle appropriée est recherchée.

Ces approches ne fonctionnent cependant que sur des images non texturées. En effet, ces méthodes ne permettent pas de reproduire une texture car seule la structure est prise en compte (voir figure 2.5). Chaque méthode possède ses avantages : les méthodes basées PDE sont très efficaces dans des situations où les textures d'arrière-plan sont uniformes et ce, même si on veut enlever de multiples petites zones. Les méthodes basées quant à elles sur la TV fonctionnent bien si la zone à enlever est relativement petite ou si la partie manquante est sensée être une structure ou une texture simple alors que les méthodes basées pixels fonctionnent correctement sur de plus grandes zones [Buy+15] (voir figure 2.5)). Notons enfin Wu et Ruan [WR08] qui améliorent l'approche TV pour prendre en compte les textures en proposant un modèle mixte mélangeant une partie structurelle et une partie texturale.

FIGURE 2.4 – Exemple de complétion par une approche TV.

2.3.2 Les méthodes basées sur la redondance

Dans cette sous-section, nous traitons de l'*inpainting* d'images via une représentation éparse. La représentation éparse d'un signal consiste à utiliser un dictionnaire constitué d'éléments élémentaires appelés des atomes. La combinaison de ces atomes permet de construire des signaux complexes. Ici, l'image I est considérée comme éparse selon un dictionnaire D et une représentation R (voir figure 2.6). Ce dictionnaire peut être initialisé en apprenant directement de la zone connue $O\backslash\Omega$. Il va représenter le contenu de l'image comme la texture et les contours. L'idée des méthodes de cette catégorie consiste à compléter la zone masquée Ω en cherchant une représentation de Ω selon le dictionnaire D calculé ci-dessus.

ELAD *et al.* [Ela+05] proposent pour cela d'utiliser l'analyse des composants morphologiques (*morphological component analysis (MCA)*). CHAN *et al.* [CSZ06] proposent une méthode qui associe la variation totale à un modèle de représentation par vaguelettes. CAI *et al.* [CCS08] utilisent quant à eux des *framelets* pour minimiser une fonctionnelle.

Définition 2.3.1. Soit $X = [x_1, \ldots, x_K], x_i \in \mathbb{R}^d$ un ensemble de données. Soit $D = [d_1, \ldots, d_n] \in \mathbb{R}^{d\times n}$ et $R = [r_1, \ldots, r_K], r_i \in \mathbb{R}^n$. L'apprentissage par dictionnaire consiste à minimiser $\|X - DR\|_F^2$. On dit que D est un dictionnaire et R une représentation. Un élément de D est appelé *atome*.

Définition 2.3.2. Soit $L_2(\mathbb{R}^2)$ l'ensemble des fonctions de carré intégrable. On appelle $X(\Phi)$ ondelette (ou *wavelet*) une collection de dilatations et de translations d'un sous-ensemble de $\Phi \subset L_2(\mathbb{R}^2)$:

$$X(\Phi) = \left\{2^{k/2}\phi(2^k x - j), \phi \in \Phi, k, j \in \mathbb{Z}\right\} \tag{2.13}$$

Si pour $\forall f \in L_2(\mathbb{R}^2)$ on a $f = \sum_{\phi\in X(\Phi)} \langle f, \phi\rangle \phi$, alors les éléments ϕ de $X(\Phi)$ sont appelés des *framelets*. Notons que la famille $X(\Phi)$ est génératrice mais n'est pas une base orthonormée. Elle en est une généralisation qui apporte de la redondance ce qui est pratique dans le cadre de domaines comme le dé-bruitage.

MAIRAL *et al.* [MES08] abordent le problème des dictionnaires d'apprentissage pour les images couleur et étendent l'algorithme de restauration des niveaux de gris basé sur la décomposition en valeurs singulières.

Ces méthodes donnent des résultats très convaincants dans le cas où la texture possède peu de structure dominante. Elles vont cependant peiner à produire des résultats

FIGURE 2.5 – Avantages et limites des deux grandes catégories d'*inpainting* : les basées diffusion et les basées *patchs*. Nous constatons que la première zone masquée (première ligne, image de gauche) a été mieux complétée par une approche de diffusion (deuxième ligne, image de gauche) que l'approche basée *patchs* (troisième ligne, image de gauche). À l'inverse, dans le cas de la deuxième zone masquée (première ligne, image de droite), c'est l'approche basée *patchs* (troisième ligne, image de droite) qui réussit mieux à compléter que l'approche de diffusion (deuxième ligne, image de droite).

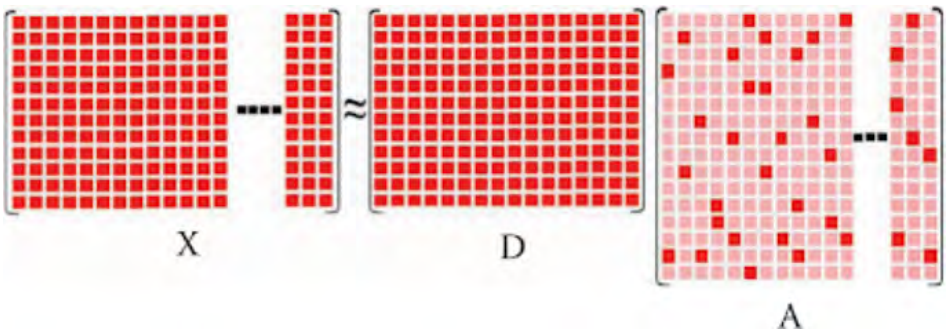


FIGURE 2.6 – Principe de la représentation éparse par un dictionnaire D (figure extraite du livre *Encyclopedia of Image Processing* [Lap18]).

FIGURE 2.7 – Représentation d'un *patch* dans une image. Le *patch* est colorié en rouge et le pixel central est colorié en jaune.

convenables dans le cas de macro-textures où il y a de fortes structures et de la répétition de motifs.

2.3.3 Les méthodes basées *patchs*

Dans cette sous-section, nous présentons la dernière catégorie d'approches d'*inpainting* orientées sur l'analyse et le traitement de *patchs*.

Dans cette famille d'approche, la complétion se fait en copiant des *patchs* de $O\backslash\Omega$ dans Ω qui sont choisis par une fonction de similarité. Ici f_I peut-être considérée comme une fonction de correspondances qui va lier les pixels de Ω aux pixels connus de $O\backslash\Omega$.

Définition 2.3.3. On appelle *patch* une sous-image Ψ d'une image I de forme carrée. Un *patch* est défini par sa demi-taille T et le pixel p situé au centre du *patch*.

Dans le cas de l'*inpainting*, les *patchs* sont de même taille mais doivent être suffisamment grands pour être du même ordre de grandeur que l'élément de texture le plus grand [EL99].

Définition 2.3.4. On appelle *fonction de similarité*, notée sim, une métrique qui mesure la distance entre deux *patchs* :

$$\text{sim} : O^2 \to \mathbb{R}^+ \tag{2.14}$$

$$(\Psi_p, \Psi_q) \mapsto \text{sim}(\Psi_p, \Psi_q) \tag{2.15}$$

Remarque 2.3.1. En pratique, la Somme des Différences au Carré (*Sum of Squared Differences* ou SSD)) est généralement utilisée comme fonction de similarité pour sa simplicité et surtout pour son efficacité en terme de temps de calcul :

$$\text{sim}(\Psi_p, \Psi_q) = \sum_{\delta \in [\![-T,T\,]\!] \times [\![-T,T]\!]} \| I(p+\delta) - I(q+\delta) \|^2 \tag{2.16}$$

D'autres métriques ont été proposées et sont souvent couplées à la *SSD*. Il y a par exemple la distance de Bhattacharya [MEG13] ou la distance de Hellinger [MRB12]. Cependant, il n'est pas clairement établi [Buy+15] qu'utiliser ces distances avec la *SSD* n'améliore sensiblement les résultats dans des cas génériques.

Les approches gloutonnes

Une première sous-catégorie de mise en correspondance de *patch* sont les approches dites locales ou gloutonnes. Elles consistent à compléter progressivement la zone masquée Ω en partant de la frontière intérieure $\partial\Omega_i$. Pour cela, elles commencent par sélectionner un *patch* Ψ_p de la frontière (c'est-à-dire centré en un pixel p de la frontière intérieure $\partial\Omega_i$). On le nomme *patch cible*. Donc une partie des pixels sont connus car appartenant à Ω tandis que d'autres sont non connus car appartenant $O\backslash\Omega$. Une recherche est effectuée pour trouver dans une fenêtre $F \subset O\backslash\Omega$ un *patch candidat* Ψ_q dont tous les pixels appartiennent à Ω et qui minimiserait la similarité. Le calcul de la similarité se fait uniquement entre les pixels connus d'où la ré-écriture de la fonction de similarité de l'équation (2.16) dans ce cas :

$$\mathrm{sim}(\Psi_p, \Psi_q) = \sum_{\delta\in[|-T,T|]\times[|-T,T|]} \mathbb{1}_{(p+\delta)\in O\backslash\Omega} \parallel I(p+\delta) - I(q+\delta) \parallel^2 \tag{2.17}$$

Une fois ceci fait, les pixels du *patch* cible qui appartiennent au masque sont remplacés relativement par ceux du *patch* candidat. Un autre pixel de la frontière est ensuite sélectionné et son *patch* associé complété jusqu'à complétion du masque. Tout l'enjeu de cette catégorie revient à déterminer dans quel ordre considérer les patch cibles pour effectuer la complétion.

L'approche de Criminisi *et al.* [CPT03] est la première approche de cette catégorie. Pour déterminer quel patch cible choisir, ils proposent un calcul de priorité. Cette priorité est le produit entre deux termes :

$$P(p) = C(p)D(p). \tag{2.18}$$

Le premier, nommée C et calculée au centre p du patch, est un score de *confiance* du patch cible qui mesure la quantité d'information présente dans le *patch* Ψ_p. Le deuxième, nommée D et calculée également au centre p, est un terme de **données** qui prend en compte la présence de structure dans le *patch* Ψ_p :

$$C(p) = \frac{\sum_{q\in\Psi_p\cap(O\backslash\Omega)} C(q)}{\mathrm{card}(\Psi_p)} \qquad D(p) = \frac{|\nabla I_p^{\perp} \cdot \overrightarrow{n_p}|}{\alpha} \tag{2.19}$$

où $\mathrm{card}(\Psi_p)$ est le nombre de pixels q présents dans le patch Ψ_p, $\nabla I_p^{\perp}$ est le vecteur perpendiculaire au gradient en p, $\overrightarrow{n_p}$ est le vecteur normal au masque Ω en p, α est un facteur de normalisation (par exemple, $\alpha = 255$ pour une image en niveau de gris). La figure 2.8 montre la complétion progressive de la région recouverte par Ω.

Plusieurs approches améliorent l'algorithme de base de Criminisi *et al.*. Ils apportent des changements notamment au niveau du calcul de priorité. Chez Huang et Hsiao [HH10] ce calcul de priorité est modifié pour prendre en compte la variation de la luminosité. Le Meur *et al.* [LGG11] améliorent le calcul de la priorité en basant le terme de données D sur des tenseurs. Dans les images de couleur, l'utilisation des tenseurs permet de mieux tenir compte de la géométrie de l'image que le gradient. Ils améliorent leur approche en complétant d'abord l'image en basse résolution [MEG13]. En effet, les structures majeures y sont mieux représentées. Il est plus facile alors de les propager dans le masque. Ils complètent ensuite l'image en haute-résolution en se servant du résultat basse résolution. Martínez-Noriega *et al.* [MRB12] proposent de changer D pour amplifier la priorité des *patchs* qui possèdent fortes structures. Guillemot *et al.* [Gui+13], au lieu de modifier D, proposent d'ajouter un troisième facteur E à P, appelé terme basé contour. Ce terme E accorde une plus grande priorité aux

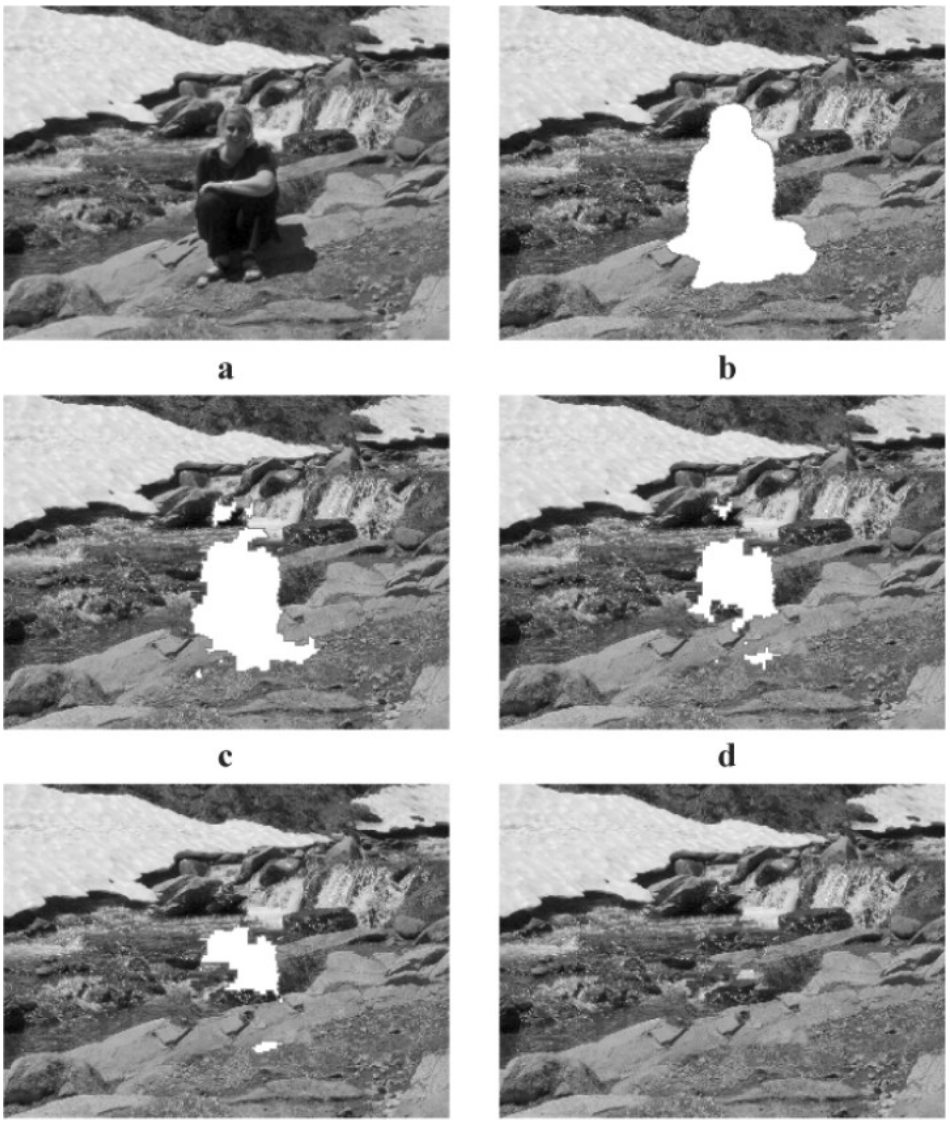


FIGURE 2.8 – Un résultat de [CPT03] qui est la première approche basée *patchs* locale. Le masque de l'image (b) est complétée progressivement en sélectionnant, pour chaque itération, le *patch* de la frontière intérieure $\partial\Omega_i$ qui a la plus haute priorité P et en le complétant avec le *patch* le plus similaire dans $O\backslash\Omega$. Les images (c) et (d) montrent cette progression. On peut constater que les *patchs* de la frontière où une structure est présente, et qui ont beaucoup de pixels déjà connus, sont complétés en priorité.

FIGURE 2.9 – Exemple de complétion par l'approche de [OH13].

patchs qui contiennent des pixels appartenant à des contours. Pour savoir si un pixel appartient à un contour, une image binaire, appelée carte de contour, est construite via un filtre de Canny. WANG *et al.* [Wan+14] proposent de modifier le terme de confiance C pour ajouter un terme de régularisation R qui va permettre de mieux contrôler le terme de confiance des pixels situés au centre du masque. Ils font également la remarque que le *patch* candidat sélectionné selon la distance de similarité dans la zone connue pour compléter le *patch* du masque n'est pas forcément le meilleur visuellement. Ils proposent alors de garder les *patchs* candidats les plus proches selon la similarité et de faire un deuxième tour parmi ces candidats pour sélectionner le plus proche selon une autre distance, la *Normalized Cross Correlation* (NCC). Pour réduire les artefacts qui peuvent subvenir après avoir copié des *patchs* dans Ω, DAISY *et al.* [DTL13] proposent de mélanger les *patchs* qui se superposent à un endroit du masque Ω durant la copie par une fonction de pondération. Un peu à la frontière avec les approches précédentes, OGAWA et HASEYAMA [OH13] propose de compléter la zone masquée par une représentation éparse des *patchs* via un dictionnaire. L'approche est divisée en deux étapes. Premièrement, ils génèrent un dictionnaire à partir des données connues de l'image. Cependant, les atomes sont sélectionnés via le critère appelé index de similarité de structure (*SSIM*) au lieu de l'équation de la Définition 2.3.1. Le *SSIM* est une mesure de qualité appliquée dans plusieurs domaines du traitement de l'image [Wan+04]. Deuxièmement, le masque est rempli progressivement en sélectionnant un *patch* de la frontière du masque via un calcul de priorité. Ce dernier est alors reconstruit en cherchant la meilleure représentation selon le dictionnaire construit dans la première étape. La figure 2.9 montre un exemple de complétion par cette approche.

Comme le résultat final va dépendre de l'ordre des *patch* à compléter, un changement mineur dans la priorité peut conduire à un résultat différent comme montrée dans la figure 2.10. De plus, en changeant la taille du *patch*, le résultat va aussi fortement varier. Le but des approches évoquées est d'arriver donc à rendre robuste l'effet de la priorité en tenant notamment compte des structures.

Les approches de minimisation d'énergie

La deuxième catégorie est celle des approches de minimisation d'énergie ou globales. Elles consistent à analyser les *patchs* de la partie connue pour ensuite compléter la zone masquée. Pour cela, elles effectuent une minimisation d'énergie globale représentant la

FIGURE 2.10 – Complétion par une méthode basée *patch* locale en fonction de la taille du patch. De gauche à droite et de bas en haut, nous avons des *patchs* de taille 7, 11 et 15 pixels.

cohérence de la complétion. Elles requièrent aussi plusieurs d'itérations pour atteindre une convergence et ainsi raffiner la partie masquée.

SUN *et al.* [Sun+05] proposent de séparer la propagation de la texture de la propagation de la structure. Pour cela, ils invitent d'abord l'utilisateur à indiquer les structures incomplètes en les traçant. Ensuite, chaque pixel appartenant au tracé et au masque Ω reçoit un label correspondant à un pixel connu du tracé ; l'attribution du label se fait par une énergie à minimiser. La texture est ensuite complétée par une approche gloutonne [CPT03] de la sous-section précédente.

WEXLER *et al.* [WSI07] proposent une complétion en calculant, pour un pixel du masque, une moyenne pondérée des pixels provenant de plusieurs *patchs* qui se superposent. Ils définissent également une fonction de cohérence globale qui mesure, pour chaque pixel, la similarité entre les *patchs* qui le recouvrent et les *patchs* de la zone connue :

$$\text{Coherence} = \prod_{p \in \Omega} \max_{q \in O \setminus \Omega} \text{sim}_g \left(\Psi_p, \Psi_q \right), \tag{2.20}$$

où sim_g est une fonction de similarité couplée à un noyau gaussien

$$\text{sim}_g(\Psi_p, \Psi_q) = \exp\left(-\frac{\text{sim}(\Psi_p, \Psi_q)}{2\sigma^2} \right), \tag{2.21}$$

où σ est l'écart-type du noyau gaussien et sim une fonction de similarité. L'intérêt de cette méthode multi-résolutions est d'utiliser une image construite à une résolution donnée comme initialisation pour la résolution supérieure. Cette approche couplée avec l'algorithme *PatchMatch* est l'approche classique d'*inpainting* basée *PatchMatch* et notamment utilisée dans le procédé *Content-aware fill* de Photoshop. Elle sera notamment détaillée dans la section 2.4.

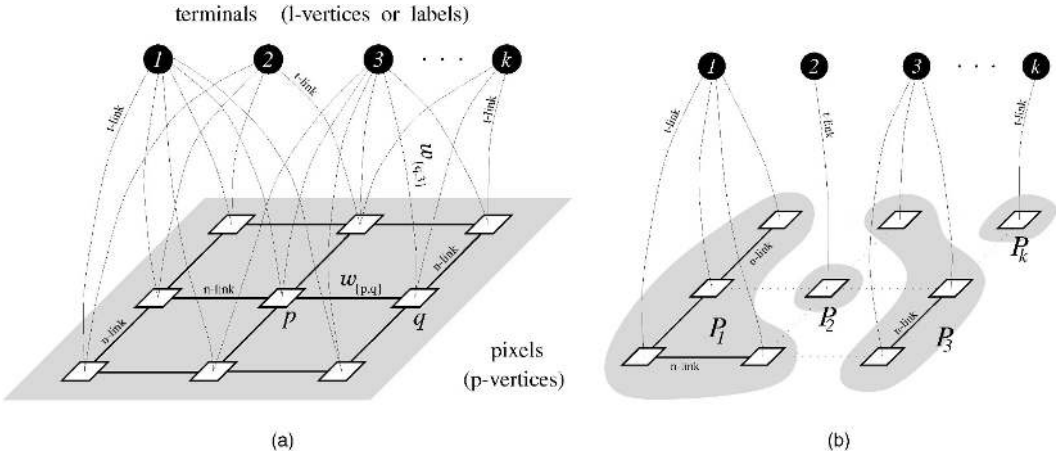


FIGURE 2.11 – Exemple d'une coupure de graphe dans le cas d'une image (figure extraite de [BK04]). Les pixels p sont représentés par des carrés blancs. Chaque pixel est lié à ses quatre voisins et à chaque terminal. Le graphe d'origine montré en (a) est coupé selon une coupe en (b). Chaque pixel n'est relié qu'à un terminal et aux pixels ayant le même terminal.

HUANG *et al.* [Hua+13] proposent d'ailleurs de modifier l'algorithme *PatchMatch* pour tenir compte d'éléments de structure indiqués par l'utilisateur. HE et SUN [HS12b] proposent d'analyser la statistique des translations entre les *patchs* de forte similarité. Cette analyse est ensuite utilisée pour effectuer un photomontage : l'image de départ est translaté selon les principaux décalages. Les images décalées sont ensuite empilées et découpées via une coupure de graphe (voir figure 2.11 et figure 2.12). Elle est détaillée dans la section 2.4 également.

Définition 2.3.5. Soit $G = \{V, E\}$ un graphe pondéré avec n nœuds distincts appelés terminaux. Une coupe $C \in E$ est un ensemble d'arêtes tel que les terminaux soient séparés dans le graphe inclus $G(C) = \{V, E - C\}$. En plus, aucun sous-ensemble de C ne sépare les terminaux dans $G(C)$.
Le coût de la coupe C, notée $|C|$, est égale à la somme des poids de ses nœuds.
Le problème de coupe minimum est de trouver parmi toutes les coupes possibles séparant les n terminaux celle qui a le coût le plus faible.

Dans cette situation, les terminaux correspondent aux différents décalages retenus. À la fin de la coupure, chaque pixel masquée sera attribué à un terminal. Il suffira alors de récupérer la valeur dans l'image décalée correspondante.

LIU et CASELLES [LC13] améliore l'approche ci-dessus en effectuant la coupure de graphe sur plusieurs résolutions qui permet d'accélérer l'exécution de l'algorithme. Cependant, pour éviter de perdre des informations de structure et de texture en basse résolution, ils proposent une manière de représenter les caractéristiques de l'image présents en résolution d'origine. Cela permet d'améliorer la précision de la correspondance quand le traitement est effectué en basse résolution. RUZIC et PIZURICA [RP15] améliorent la recherche d'un candidat en proposant un *framework* qui utilise des descripteurs contextuels (comme les réponses au filtre de Gabor) ainsi qu'une optimisation concernant l'*inpainting* qui utilise les champs de Markov aléatoire. Enfin NEWSON *et al.* [New+17] et ZHIDAN LI [Zhi18] s'intéresse à l'initialisation de l'énergie et aux nombre d'étages dans la pyramide multi-résolutions. L'idée intéressante est de fournir une initialisation plus cohérente avec la structure de l'image avant de minimiser l'énergie.

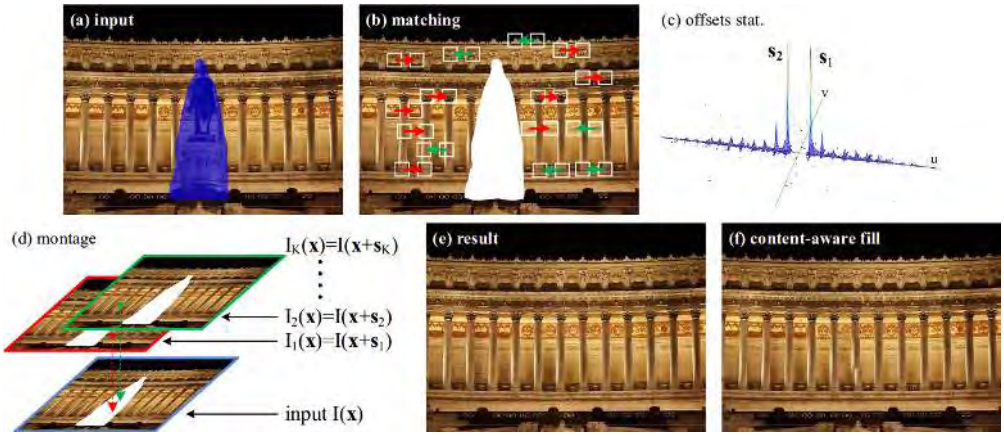


FIGURE 2.12 – Complétion par analyse statistique des décalages (figure extraite de [HS12b]). (a) : image d'entrée. (b) : recherche des patchs similaires dans la zone connue. (c) : analyse statistique des patchs similaires, seuls les décalages dominants sont retenus. (d) : Combinaison des images décalées générées à partir des décalages dominants. (e) : le résultat de l'approche. (f) : la méthode *Content-aware fill* de Photoshop.

2.3.4 Les méthodes par apprentissage

Ces méthodes très récentes se servent des avancées en apprentissage profond [Sch15] pour créer un modèle d'apprentissage capable de compléter une zone d'une image. Comme toutes les méthodes d'apprentissage, il y a une phase d'apprentissage d'un modèle suivie d'une phase d'application du modèle.

SONG *et al.* [Son+18] proposent de diviser le processus en trois étapes (voir figure 2.13) : inférence, association et translation. Dans la première étape, un réseau est entraîné pour initialiser la région masquée avec des prédictions grossières. Ces prédictions sont enregistrées dans un objet appelé carte de caractéristiques. Dans la deuxième étape, chaque neurone qui est dans la zone masquée est associé à un neurone connu de la carte de caractéristiques. Dans la troisième étape, un deuxième réseau est entraîné pour permettre de transformer les caractéristiques trouvées en une image complète.

WANG *et al.* [Wan+18] proposent de combiner un champ de Markov aléatoire à la phase où les caractéristiques associés aux neurones masqués sont utilisés pour reconstruire l'image. Cela permet d'éviter d'avoir des artefacts vu que le champ de Markov aléatoire agit comme un terme de régularisation. Ils ajoutent aussi une fonction de coût orientée reconstruction qui contraint la génération des données dans le masque par rapport à l'origine des données copiées. Ainsi les données proches du masque auront plus de chance d'être utilisées que celles éloignées du masque.

Alors que les deux approches ci-dessus se basent sur les Réseaux de Neurones Convolutifs (*Convolutional Neural Network (CNN)*, ZHANG *et al.* [Zha+18] proposent un modèle qui se rapproche des Réseaux Antagonistes Génératifs (*Generative Adversarial Networks* (GAN)) (voir figure 2.14). Les GAN sont utilisés pour générer des images très réalistes. Les méthodes qui les utilisent utilisent généralement deux réseaux qui sont mis en compétition. Un premier réseau, dit génératif, apprend le modèle de génération des données en entrée. Le deuxième réseau, dit discriminatif, essaie de discriminer les données réelles en entrée des données générées par le premier réseau. Le but est de "tromper" le réseau discriminatif, c'est-à-dire de maximiser l'erreur de classification du deuxième réseau.

L'avantage de ces méthodes est de pouvoir entraîner le modèle sur un système avec

une forte capacité de calcul et de l'utiliser ensuite dans un environnement moins contraint (par exemple, sur un smartphone). Bien que ces méthodes semblent prometteuses, elles ne sont pas applicables dans notre contexte. En effet, si une approche par apprentissage est généralisable, il faut la spécialiser pour une catégorie d'image.

Ces approches ne font que de la généralisation locale [Cho17], c'est-à-dire que l'on ne peut adapter qu'à de nouvelles situations qui soient très proches des données entraînées (*cf.* figure 2.15). De plus, leur arrivée tardive dans le cadre de la thèse ne permet de les analyser et de les tester en profondeur.

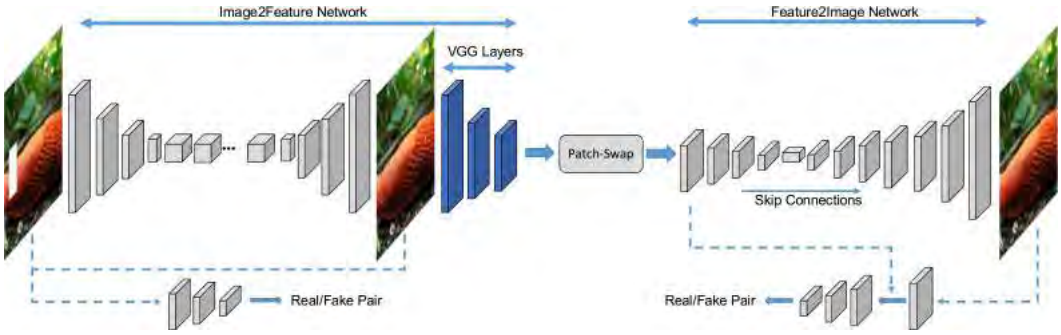


FIGURE 2.13 – Illustration du processus d'*inpainting* via deux réseaux (figure extraite de [Son+18]. Un premier réseau (*Image2Feature*) est utilisé pour deviner de manière grossière la zone masquée ; une carte de caractéristique en est extraite. Ensuite, chaque neurone de la zone masquée dans la carte de caractéristique est associé à un neurone connu. Enfin, un autre réseau (*Feature2Image*) est utilisé pour traduire la carte de caractéristique en une image complète.

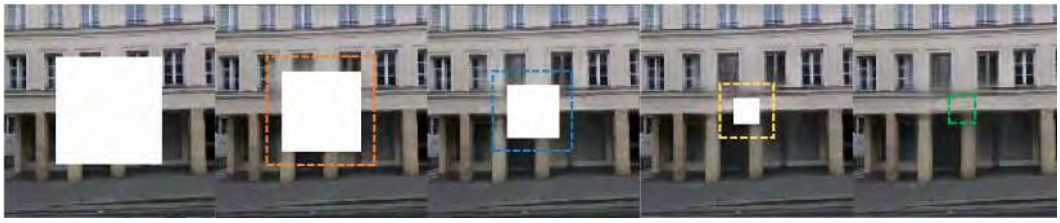


FIGURE 2.14 – Résultats de l'approche de [Zha+18].

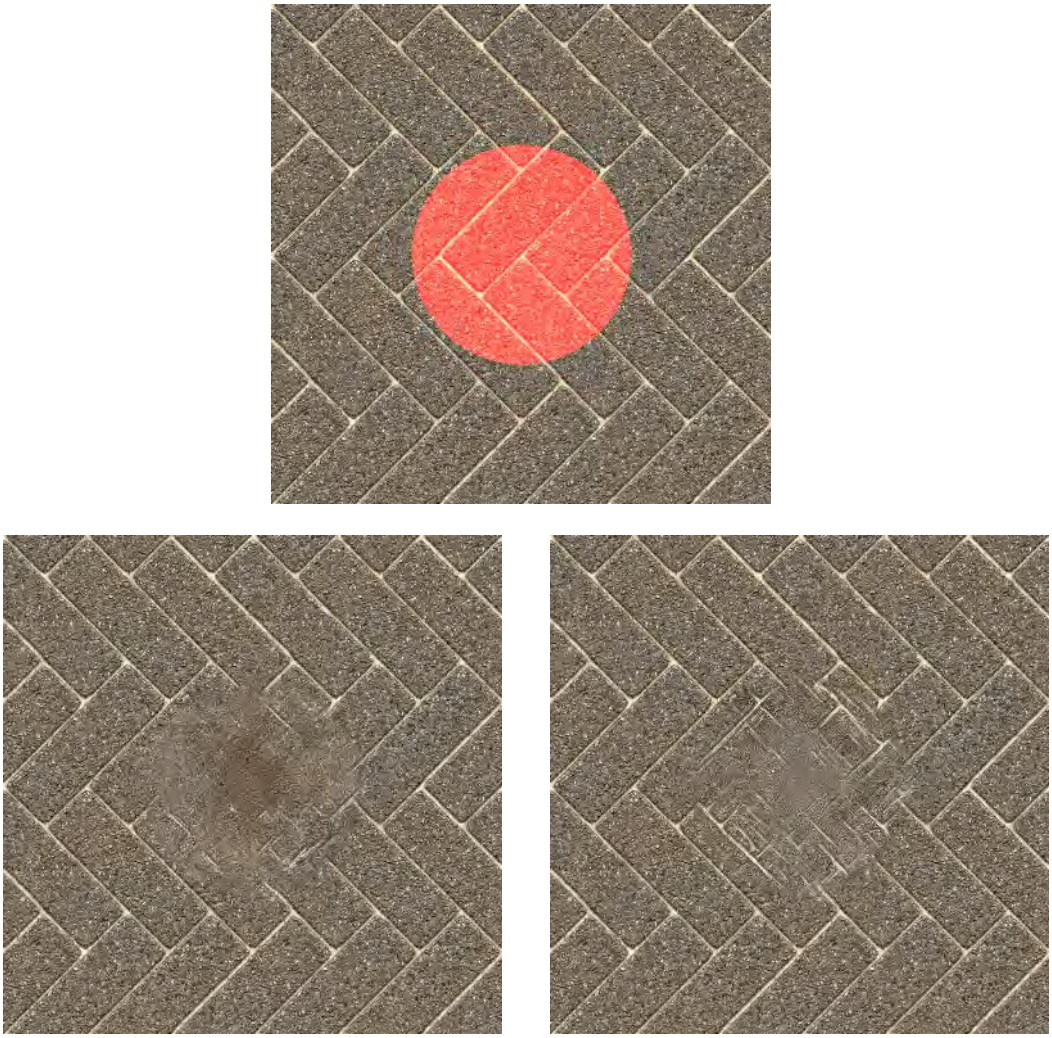


FIGURE 2.15 – Limites d'une approche par apprentissage appliqué à une texture (zone à compléter coloriée en rouge). Les résultats proviennent de l'approche de YU *et al.* [Yu+18]. Nous avons utilisé les modèles pré-entraînés ImageNet (gauche) et Place2 (droite) pour créer ces résultats. Ces deux modèles sont entraînés pour les images naturelles et pour les objets. Bien qu'il soit normal que le premier résultat soit décevant, le deuxième devait être meilleur.

2.4 Analyse des méthodes d'*inpainting* par *PatchMatch* et par *offsets*

Nous présentons plus en détails deux approches basées *patchs* car elles servent de base pour nos contributions. Ces deux approches donnent généralement de bons résultats sur des textures d'intérieur et sont peu coûteuses et de ce fait peuvent être mises en oeuvre sur des systèmes mobiles dont les capacités de calcul sont parfois limitées.

2.4.1 Approche par *PatchMatch*

L'approche par *inpainting* [WSI07] basé sur *PatchMatch* consiste à utiliser l'algorithme de recherche de correspondances de même nom [Bar+09] pour chercher à établir une carte de correspondances c entre les pixels de Ω et les pixels de $O\backslash\Omega$. Comme cette approche est multi-résolutions, l'idée générale est de trouver une correspondance entre la zone masquée et la zone connue en basse résolution et de l'affiner dans les résolutions supérieures grâce à *PatchMatch*. Depuis, plusieurs approches évoquées dans la section 2.3 [CT14 ; Mor+12 ; Hua+13] ont contribué à améliorer cette méthode.

Dans la suite, nous allons d'abord expliquer l'algorithme de base (dans le cas de deux images distincts). Puis nous allons détailler son application dans le case de l'*inpainting*.

Algorithme de base

Nous commençons par décrire l'algorithme de base [Bar+09] qui sert pour la correspondance entre les pixels du masque et le reste de l'image.

Définition 2.4.1. Soient deux images I_1 et I_2 définies respectivement sur les domaines $U \subset \mathbb{N}^2$ et $V \subset \mathbb{N}^2$. On appelle carte de correspondance c la représentation d'une fonction injective entre une image de départ et une image d'arrivée :

$$c\colon U \to V \tag{2.22}$$
$$(x, y) \mapsto (u, v) \tag{2.23}$$

La figure 2.16 nous donne un exemple de carte de correspondance. Cet exemple a pour but d'expliquer la représentation que nous utiliserons pour les cartes de correspondance dans la suite du manuscrit.

Remarque 2.4.1. Il n'est pas nécessaire que U et V aient les mêmes dimensions. Cela signifie que l'on peut appliquer *PatchMatch* à deux images de tailles différentes.

L'algorithme *PatchMatch* a pour but de calculer une carte de correspondance entre deux images I_1 et I_2 qui associe à chaque patch de I_1 son plus proche voisin dans I_2 selon une fonction de similarité.

Définition 2.4.2. Soit une image I_1 définie sur un domaine U et I_2 une définie sur un domaine V. Soit $c\colon U \to V$ une carte de correspondance. On appelle I_1 l'*image de référence* dans *PatchMatch*. On appelle I_2 l' *image patch* dans *PatchMatch*.

L'algorithme *PatchMatch* est divisé en trois étapes (*cf*. algorithme 1) :

— **Étape d'initialisation :** Une carte de correspondances c est doit être fournie entre l'image de référence et l'image patch. Généralement elle est initialisée de manière aléatoire [BRR11]. Cependant, dans les approches multi-résolutions (comme par exemple Hu *et al*. [HSL16] pour résoudre un problème de flux optique) qui utilisent *PatchMatch*, la carte de correspondance calculée à une résolution est considérée comme une nouvelle initialisation pour les résolutions supérieures.

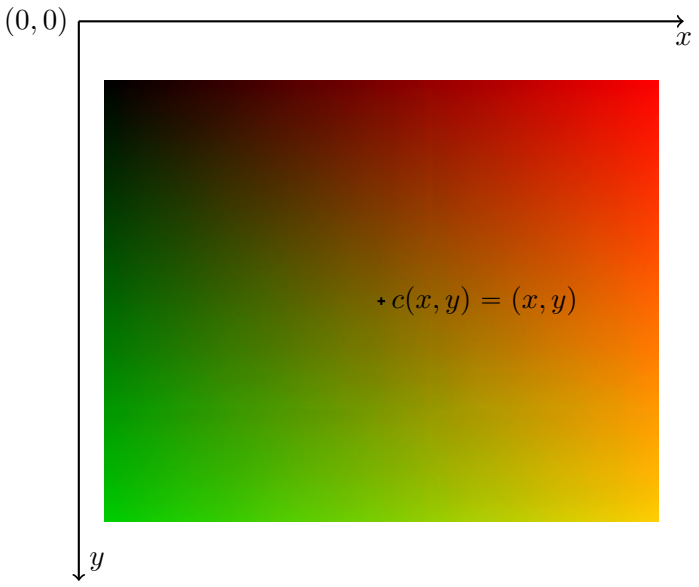


FIGURE 2.16 – Représentation d'une carte de correspondance avec $c = Id$, "la carte de correspondance qui mappe chaque pixel en lui même". Cet exemple permet d'expliquer le code de couleurs appliqué : si (x, y) est un pixel et (u, v) son image par c, u est codé sur le canal rouge et v est codé sur le canal vert. C'est pourquoi dans cet exemple on a un dégradé vers le rouge pour les x croissants $(u = x)$ et vers le vert pour les y croissants $(v = y)$.

— **Étape de propagation :** Répétée n_{iter} fois (*cf.* étape 9 de algorithme 1), cette étape consiste à regarder la correspondance du voisinage de chaque pixel. Soit $p \in O$ et un de ses voisins $p + \delta$ avec $\delta = [\pm 1, \pm 1]$. L'image de p (respectivement $p + \delta$) par c est récupérée et est notée $c(p)$ (respectivement $c(p + \delta)$). Si le score de similarité est meilleure entre p et $c(p + \delta) - \delta$ qu'entre p et $c(p)$, l'image de p par c est dorénavant $c(p + \delta) - \delta$. La figure 2.17 schématise cette étape.

— **Étape de casualisation :** . Répétée également n_{iter} fois (cf étape 13 de l'algorithme 1), cette étape consiste à sélectionner de manière aléatoire un pixel q situé dans un disque de rayon ω centré en p. Si le score de similarité est meilleure entre p et q qu'entre p et $c(p)$, l'image de p par c est dorénavant q. Au départ, ω a un rayon égal au maximum des dimensions de l'image. La recherche se fait naturellement sur l'intersection du domaine de l'image et du disque. À chaque itération de la fonction MISEÀJOURALÉATOIRE de l'algorithme 1, le rayon est divisé par 2. La figure 2.18 schématise cette étape.

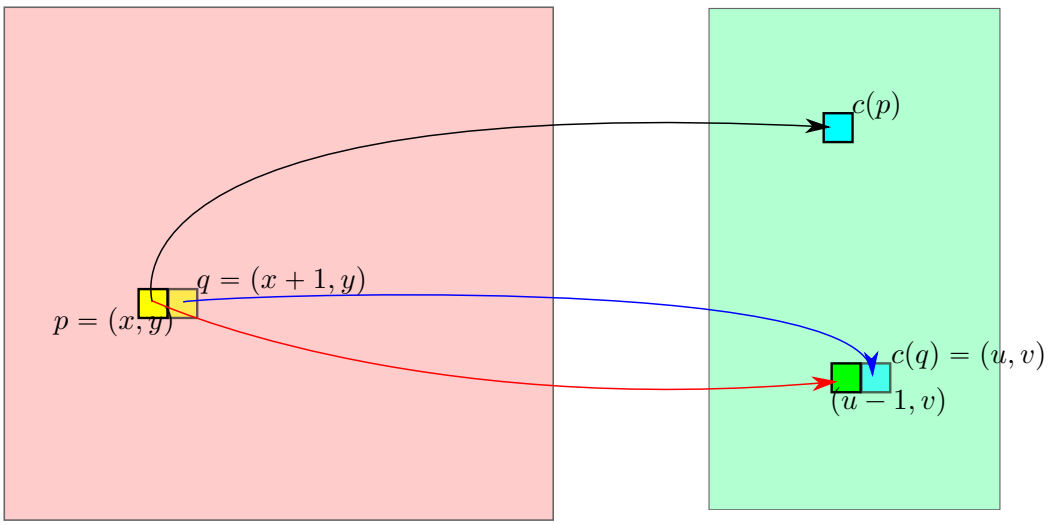


FIGURE 2.17 – Schéma décrivant l'étape de propagation. Le pixel p est associé au pixel $c(p)$ (flèche noire). Son pixel voisin de droite q est associé au pixel $c(q)$ (flèche bleue). Si la mesure de similarité entre p et le voisin de gauche de $c(q)$ (flèche rouge) est meilleure que la similarité entre p et $c(p)$, p va être associé au voisin de gauche de $c(q)$.

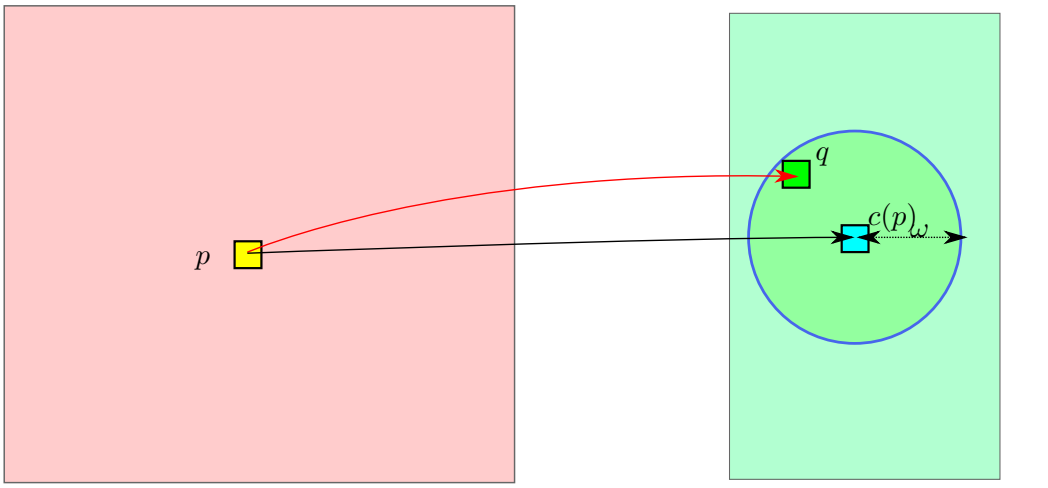


FIGURE 2.18 – Schéma décrivant l'étape de randomisation. Le pixel q est choisi de manière aléatoire dans le disque vert centré en p. La similarité étant meilleure entre les patchs centrés en p et q qu'en p et $c(p)$, q devient la nouvelle image de p par c (flèche rouge).

Algorithm 1 Algorithme *PatchMatch*

Entrée: Image I_1, Patch I_2 ▷ I_1 est l'image de référence, I_2 est l'image patch
Sortie: Carte de correspondances c

Procédure *PatchMatch*$(I_1, I_2, n_{\text{iter}}, c_{\text{init}}, \omega)$
  Si $c_{\text{init}} \neq \varnothing$ **Alors** ▷ Étape d'initialisation
    $c \leftarrow c_{\text{init}}$
  Sinon
    $c \leftarrow$ ALÉA( )
  Fin Si
  Pour $j \leftarrow 1 : n_{\text{iter}}$ **Faire**
    Pour tout $p \in I_1$ **Faire**
      MiseÀJourVoisins$(p, -1, 0)$ ▷ Étape de propagation
      MiseÀJourVoisins$(p, 1, 0)$
      MiseÀJourVoisins$(p, 0, -1)$
      MiseÀJourVoisins$(p, 0, 1)$
      MiseÀJourAléatoire(p, ω) ▷ Étape de randomisation
    Fin Pour
  Fin Pour
Fin Procédure

Fonction MiseÀJourVoisins(p, dx, dy)
  $\delta \leftarrow (dx, dy)$
  $q \leftarrow c(p + \delta)$
  Si $\text{SSD}(p, q - \delta) \leqslant \text{SSD}(p, c(p))$ **Alors**
    $c(p) \leftarrow q - \delta$
  Fin Si
Fin Fonction

Fonction MiseÀJourAléatoire(p, ω)
  Pour $j \leftarrow 1 : n_{\text{iter}}$ **Faire**
    $\delta \leftarrow (\text{ALÉA}(1, 2^{-j}\omega), \text{ALÉA}(1, 2^{-j}\omega))$
    $q \leftarrow c(p) + \delta$
    Si $\text{SSD}(p, q) \leqslant \text{SSD}(p, c(p))$ **Alors**
      $c(p) \leftarrow q$
    Fin Si
  Fin Pour
Fin Fonction

La convergence est garantie et expliquée dans le papier de BARNES *et al.* [Bar+09]. Cela vient du fait qu'un pixel aura certainement sa meilleure image par c par l'initialisation aléatoire. Ainsi son voisinage aura une image « correcte » en une itération.

La figure 2.19 montre un exemple d'application de l'algorithme dans le cas de la recherche de correspondance entre deux images.

Complétion via *PatchMatch*

L'application de *PatchMatch* dans le processus d'*inpainting* (*cf.* algorithme 2) est détaillée dans ce paragraphe. La figure 2.20 donne une idée globale du processus.

Soit I une image et Ω un masque. L'idée est de considérer du point de vue de *PatchMatch* $I_{|\Omega}$ comme l'image de référence et $I_{|O\setminus\Omega}$ comme l'image patch. À chaque pixel

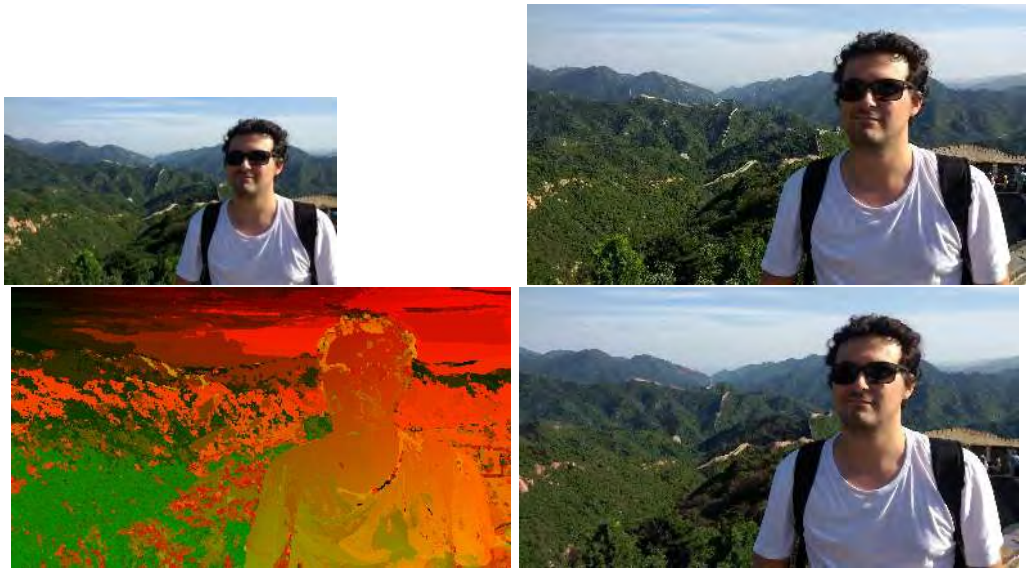


FIGURE 2.19 – Exemple de recherche d'une carte de correspondance avec *PatchMatch*. Nous cherchons une correspondance entre une image (haut-droite) et une image *patch*, de taille plus petite (haut-gauche). La taille des *patchs* est la même entre les deux images, même si elles sont de taille différente. La carte de correspondance obtenue (bas-gauche) nous permet de recréer une image (bas-droite) ressemblant à l'image d'origine où la valeur chaque pixel a été remplacée par celle du pixel correspondant. Le choix de la taille T du *patch* est fait de la même manière que dans la sous-section 2.3.3. De manière empirique, une taille 9×9 donne des résultats satisfaisants.

de Ω est attribué un pixel de $O \backslash \Omega$ de sorte que les patchs centrés en ces deux pixels minimisent une distance de similarité.
Or $I_{|\Omega}$ ne possède pas de valeurs valables. Une première étape consiste à initialiser la zone masquée pour pouvoir calculer une première estimation de c. Pour cela, une approche d'*inpainting* basée diffusion [CS01a] est généralement appliquée en Ω. Cela permet d'avoir des données globalement valables en basse résolution pour pouvoir comparer avec celles présentes en dehors de Ω.
Une pyramide de N résolutions est aussi construite à partir de I. Nous notons I^k l'image I au k-ième étage de la pyramide. L'ordonnancement de la pyramide est la suivante : l'étage 0 est la résolution d'origine et monter de chaque étage correspond à diviser la taille de l'image par 2. Donc l'image I^k de l'étage k de la pyramide aura une taille $\times 2^k$ plus petite que I^0. La taille T des *patchs* à l'étage k est aussi divisée par 2^k.

À l'étage N de plus faible résolution, la carte de correspondance c est initialisée de manière aléatoire et est mis à jour via *PatchMatch* (ce qui est possible vu que nous possédons une donnée valide dans Ω grâce à la diffusion). Ensuite, pour chaque étage k de la pyramide, la carte de correspondance est récupérée depuis l'étage supérieure $k+1$. La carte de correspondance c est ensuite appliquée à l'image $I^k_{|\Omega}$: cela permet de récupérer la correspondance de l'étage $k-1$. Pour cela, il est possible de lisser l'application de c en appliquant un noyau gaussien sur le voisinage de chaque pixel de Ω [WSI07] :

$$I(p) = \frac{\sum_{q \in \mathcal{V}_p} \omega_q I(c(q))}{\sum_{q \in \mathcal{V}_p} \omega_q}. \tag{2.24}$$

La carte c est ensuite raffinée en appliquant l'algorithme *PatchMatch* entre $I^k_{|\Omega}$ et $I^k_{|O \backslash \Omega}$.
La figure 2.21 montre un exemple de zone complétée après effacement du masque.

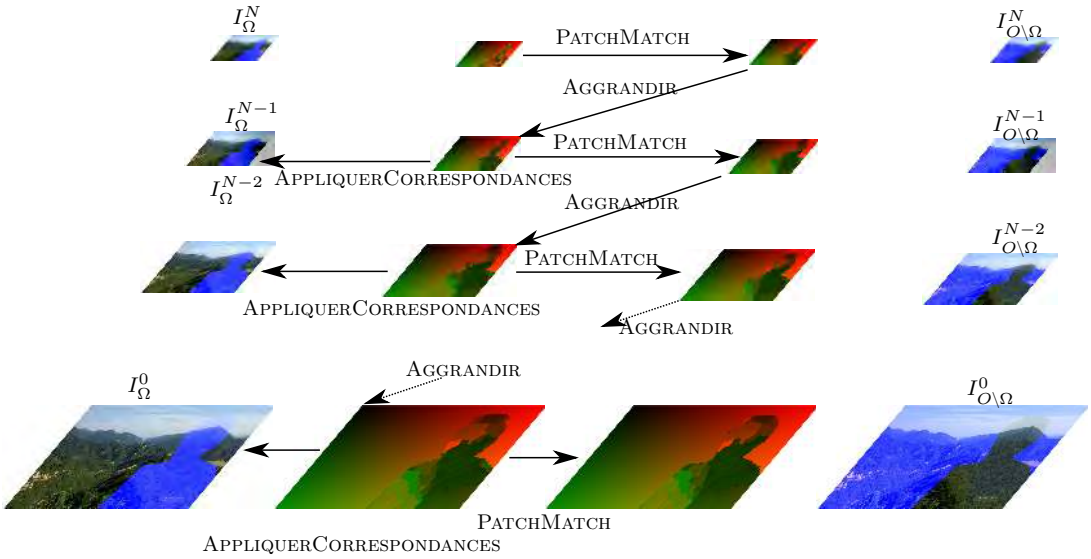


FIGURE 2.20 – Processus de l'*inpainting* basée sur *PatchMatch*. À chaque étage k de la pyramide multi-résolutions, la carte de correspondance c^k est obtenue à partir de la carte c^{k-1} de l'étage supérieur via AGGRANDIR (ou initialisée de manière aléatoire à l'étage N). Elle est appliquée à I_Ω^k via APPLIQUERCORRESPONDANCE. Elle est ensuite affinée via la fonction PATCHMATCH qui met à jour la correspondance entre I_Ω^k et $I_{O\setminus\Omega}^k$.

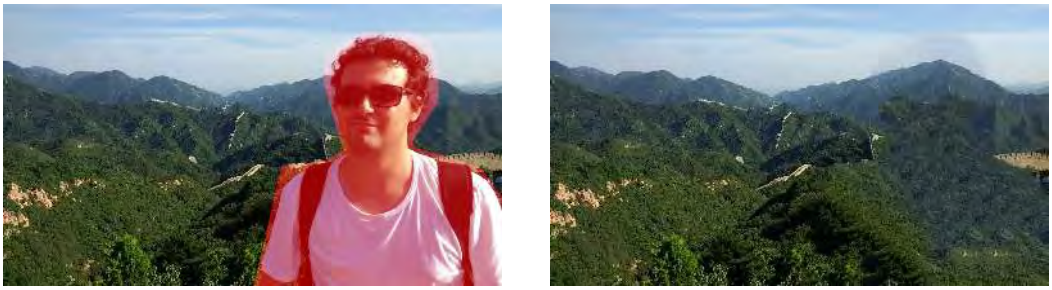


FIGURE 2.21 – Exemple d'*inpainting* en utilisant *PatchMatch*. Gauche : image avec la zone masquée en surbrillance rouge. Droite : image après traitement.

Avantages et limites L'avantage principal de *PatchMatch* est de pouvoir effacer rapidement avec un résultat convenable des éléments d'une photo. Il fonctionne très bien également pour compléter des textures avec peu de structure. Cependant, ses limites se trouvent dans le cas où la texture à compléter possède des structures fortes (carrelage, briques, voir première ligne de la figure 2.22). De plus, si une image possède deux textures différentes et que l'on doit effacer un objet à la frontière entre les deux, cette dernière sera mal reconstruite (voir deuxième ligne de la figure 2.22). Notons que certaines approches proposent des améliorations de l'*inpainting* par *PatchMatch*. CHICAN et TAMAAZOUSTI [CT14] proposent, par exemple, de contraindre la recherche de correspondance de *PatchMatch* [Bar+09] en analysant les décalages entre des points d'intérêt. MORSE *et al.* [Mor+12] proposent, dans leur contexte de trouver une correspondance entre deux images stéréo d'ajouter deux étapes. En plus du processus classique de recherche entre la zone connue et la zone masquée, une étape de stéréo-correspondance et de stéréo-propagation qui cherchent des correspondances entre deux *patchs* des deux images stéréo sont ajoutées pour ajouter, à la fin, une cohérence entre les deux images.

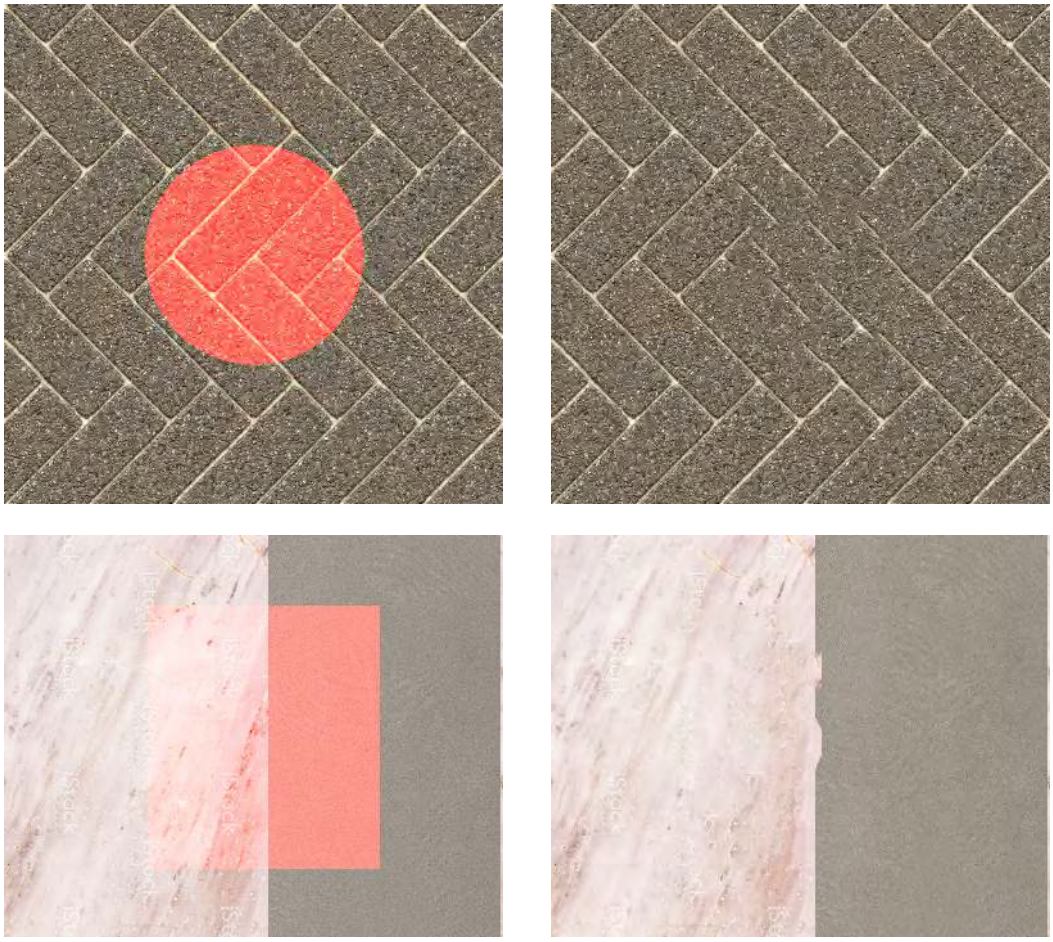


FIGURE 2.22 – Limites de *PatchMatch* : exemple sur une texture avec un motif structuré ainsi que sur image possédant deux textures.

Algorithm 2 Algorithme d'*inpainting* avec *PatchMatch*

Entrée: Image I, Masque Ω
Sortie: Image complétée I_c

Procédure INPAINT(I, Ω, N)
  Appliquer une approche d'*inpainting* par diffusion à $I_{|\Omega}$
  Créer une pyramide multi-résolutions pour I and Ω avec N étages
  Pour $i \leftarrow 1 : N$ **Faire**
    Si $i = 1$ **Alors**
      $c \leftarrow$ ALÉA( ) ▷ Premier étage, initialisation de la carte de correspondances
    Sinon
      $c \leftarrow$ AGGRANDIR$(c, 2)$ ▷ Récupérer la carte de l'étage précédent
    Fin Si
    APPLIQUERCORRESPONDANCES$(I_{|\Omega}, c)$
    Pour $j \leftarrow 1 : n_{\text{PM}}$ **Faire**
      $c \leftarrow$ PATCHMATCH$(I_{|\Omega}, I_{|O\setminus\Omega}, n_{\text{iter}}, c)$
    Fin Pour
  Fin Pour
Fin Procédure

2.4.2 Approche par analyse statistique

Nous avons vu précédemment l'efficacité de l'*inpainting* par *PatchMatch* pour compléter les masques de textures sans structure dominante. Cependant, cette approche rencontre des difficultés dès que celles-ci sont présentes. À l'inverse de *PatchMatch* qui peine dans cette situation, l'approche par analyse statistique arrive à gérer ces structures pour fournir un résultat qui est cohérent.

Dans l'approche proposée par HE et SUN [HS12b], l'objectif est d'analyser les décalages entre les points d'intérêts de la structure de la texture et d'utiliser cette analyse pour remplir la zone masquée Ω. Pour cela, une analyse statistique est effectuée en étudiant la similarité des *patchs* (sous-images centrées en un pixel) dans la zone connue $O\setminus\Omega$. Une fois l'analyse effectuée, les décalages majeures sont retenus pour remplir la zone masquée. Pour cela, le remplissage est considérée comme un problème de photomontage (détaillé par la suite).

Les *offsets* sont l'outil principal utilisé dans cette approche pour analyser la texture. Ces derniers représentent les décalages entre les pixels d'une même structure.

Définition 2.4.3. Soit une image I définie sur $O \subset \mathbb{N}^2$. On appelle *offset* un vecteur $\overrightarrow{o} \in \mathbb{Z}^2\setminus(0,0)$ qui à un pixel p associe un pixel $p + \overrightarrow{o}$ qui minimise la fonction de similarité entre les deux *patchs* centrés en ces pixels :

$$\overrightarrow{o} = \text{argmin}_{u\in\mathbb{Z}^2\setminus(0,0)} \text{sim}(p, p + \overrightarrow{u}) \tag{2.25}$$

Le processus global est présenté dans algorithme 3 :

— La procédure STATISTIQUE+GRAPHCUT est la procédure principale. L'image I et le masque M binaire représentant la zone Ω sont redimensionnées de sorte à avoir une taille plus petite avec un ratio r qui est un multiple de 2. Réduire l'image permet d'accélérer l'analyse des *offsets* et d'enlever les petites structures qui peuvent provoquer du bruit.

— On obtient la liste est *offsets* via la fonction CALCULEROFFSETS. Cette fonction est détaillée dans la sous-section 2.4.2.

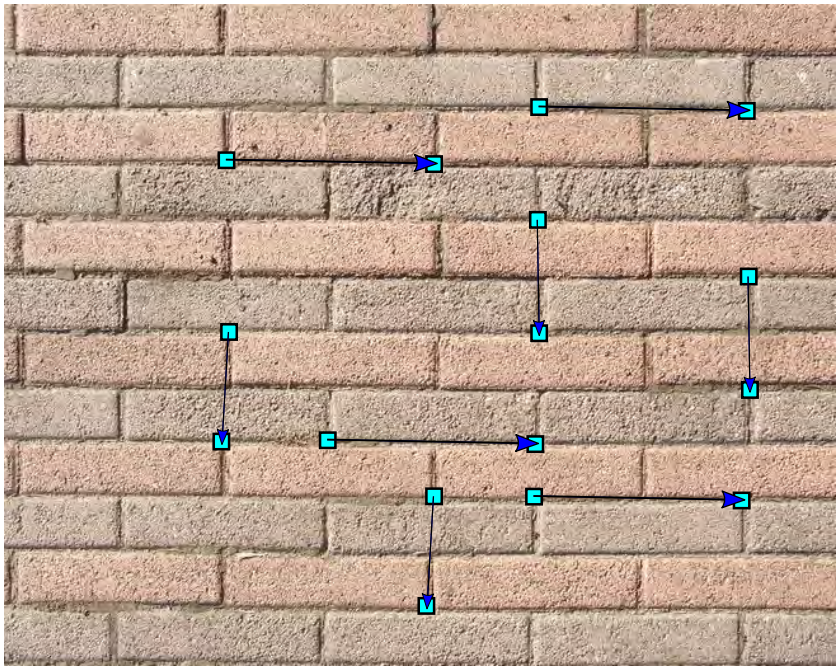


FIGURE 2.23 – Exemple d'*offsets* dans une texture

— Une fois la liste obtenue, nous l'utilisons pour attribuer un label à chaque pixel de Ω qui permet de trouver la valeur à copier grâce à CALCULERLABELS. Cette fonction est détaillée dans la sous-section 2.4.2.
— Ces labels étant calculés en basse-résolution, ils sont multipliés par le ratio r et appliqués à l'image I grâce à APPLIQUERLABELS.

Celui-ci a été amélioré par d'autres approches [Liu+18 ; Yan+16].

Calcul des *offsets*

Pour calculer les *offsets*, nous devons chercher, pour chaque *patch* de $O \backslash \Omega$, le *patch* le plus proche par une distance de similarité. Or, la recherche, pour chaque taille $2 * T$ de la zone connue de I, est très coûteuse en temps. En effet, selon la taille T, on aurait des *patchs* qui auraient $3(2T)^2$ données. En raison de la dimension élevée des données, celle-ci sont transposées dans un autre espace (noté ici F) [Xia+11] où les *patchs* auront un nombre de données fixe via une réduction analogue à l'Analyse en Composantes Principales.

Comme on passe dans un espace plus petit, il ne faut retenir que l'information pertinente pour la suite Pour cela, chaque *patch* est transposé dans l'espace de couleur YCbCr [HS12a] où Y représente la luminance et Cb/Cr représente la chrominance. L'intérêt de passer d'abord dans cet espace est le suivant : l'œil humain est plus sensible à la luminance qu'à la chrominance. Il est donc possible de dégrader la chrominance tout en conservant l'information (la luminance) qui nous intéresse pour la suite.

Ensuite chaque *patch* est projeté dans les bases représentant les matrices de Walsh-Hadamard [HH05]. Pour rappel, c'est une transformation de Fourier orthogonale. Dans cette projection, les 16 premières bases sont utilisées pour le canal Y et les 4 premières pour les canaux Cb et Cr. L'espace final F est donc un espace, fixe quelque soit la taille du *patch*, de dimension 24.

Remarque 2.4.2. Les matrices de Walsh-Hadamard ont pour taille des puissances de

2.Par conséquence, la taille $2*T$ du *patch* doit être une puissance de 2 ou $T = 2^{k-1}, k > 1$. En pratique, T prend la valeur 4.

Par la suite, on dit qu'à un *patch* de l'image I, on associe un point a de F qui représente sa projection dans cet espace.
Un arbre $k-$d [OL82] est ensuite construit sur l'ensemble des points de F afin de rechercher plus rapidement pour chaque point a de F le point b qui est le plus proche. Ainsi, à chaque point a de F est attribué un autre point $b \in F, a \neq b$, qui minimise une fonction de similarité (ici la norme $L2$). La norme n'est pas pondérée car les points de F le sont déjà par la projection via les matrices de Walsh-Hadamard (pour accorder plus d'importance à la luminance qu'à la chrominance). Une fois ce dernier trouvé, l'*offset* est obtenu en calculant la différence entre la position des pixels centraux des *patchs* représentés par les points a et b. Cependant, il est probable que le pixel candidat trouvé soit très proche du pixel traité p. Cela conduit donc à un *offset* proche du vecteur nul. Pour contrer cela, un seuil τ est établi pour filtrer le voisinage direct de p. L'équation 2.25 est donc ainsi modifiée :

$$\overrightarrow{o} = \text{argmin}_{\overrightarrow{u} \in \mathbb{Z}^2, |\overrightarrow{u}| > \tau} \text{sim}(p, p + \overrightarrow{u}). \tag{2.26}$$

Soit $\mathcal{O}$ l'ensemble des *offsets* d'une image I. Soit s la fonction qui à tout pixel p associe son *offset* $\overrightarrow{o}$:

$$s : O \to \mathcal{O}, p \mapsto s(p) = \overrightarrow{o}. \tag{2.27}$$

Un histogramme est finalement calculé sur l'ensemble des *offsets* :

$$H(\overrightarrow{o}) = \sum_{p \in O} \mathbb{1}(s(p) = \overrightarrow{o}). \tag{2.28}$$

Celui-ci représente les *offsets* qui ont au moins un pixel valide dans I. Ensuite, ne sont gardés que les K *offsets* ayant le plus grand nombre de pixels valides. Cela permet de ne conserver que les directions majeures et de réduire le temps d'exécution pour la suite. Dans l'approche de He et Sun [HS12b], K est fixé de manière empirique à 60.

La figure 2.24 montre plusieurs exemples d'images où sont affichés dans la deuxième colonne les principaux *offsets*.

Attribution des *offsets* dans la zone masquée

L'idée de cette étape est de traiter le complétion comme un photo-montage [Aga+04]. Pour rappel, on génère pour chaque *offset* $\overrightarrow{o}$ une image $I_{\overrightarrow{o}}$ qui correspond à l'image I « décalée » selon l'*offset* $\overrightarrow{o}$. On sélectionne ensuite pour chaque image décalée $I_{\overrightarrow{o}}$ une partie qui sera collée dans le masque Ω de I. Par ailleurs, ce « copier-coller » doit former une partition : à chaque pixel de Ω n'est copiée la valeur que d'une seule image $I_{\overrightarrow{o}}$.
Pour effectuer ce photo-montage, chaque *offset* $\overrightarrow{o}$ est associé un label l.

Soit L la fonction qui associe à tout *offset* un label l :

$$L : \mathcal{O} \to [|0, N|[\tag{2.29}$$

$$\overrightarrow{o} \mapsto l = L(\overrightarrow{o}). \tag{2.30}$$

Dire qu'un pixel p a pour label $l = L(\overrightarrow{o})$ signifie que l'information présente au pixel $p + \overrightarrow{o}$ (ou au pixel p de $I_{\overrightarrow{o}}$) est récupérée et copiée en p.

L'objectif de cette étape est d'attribuer pour chaque pixel $p \in \Omega$ un label l correspondant à un *offset* en minimisant l'énergie suivante :

$$E = \sum_{p \in \overline{\Omega}} E_d(p, l) + \sum_{(p_1, p_2) | p_1 \in \Omega, p_2 \in \Omega} E_r(p_1, p_2, l_1, l_2), \tag{2.31}$$

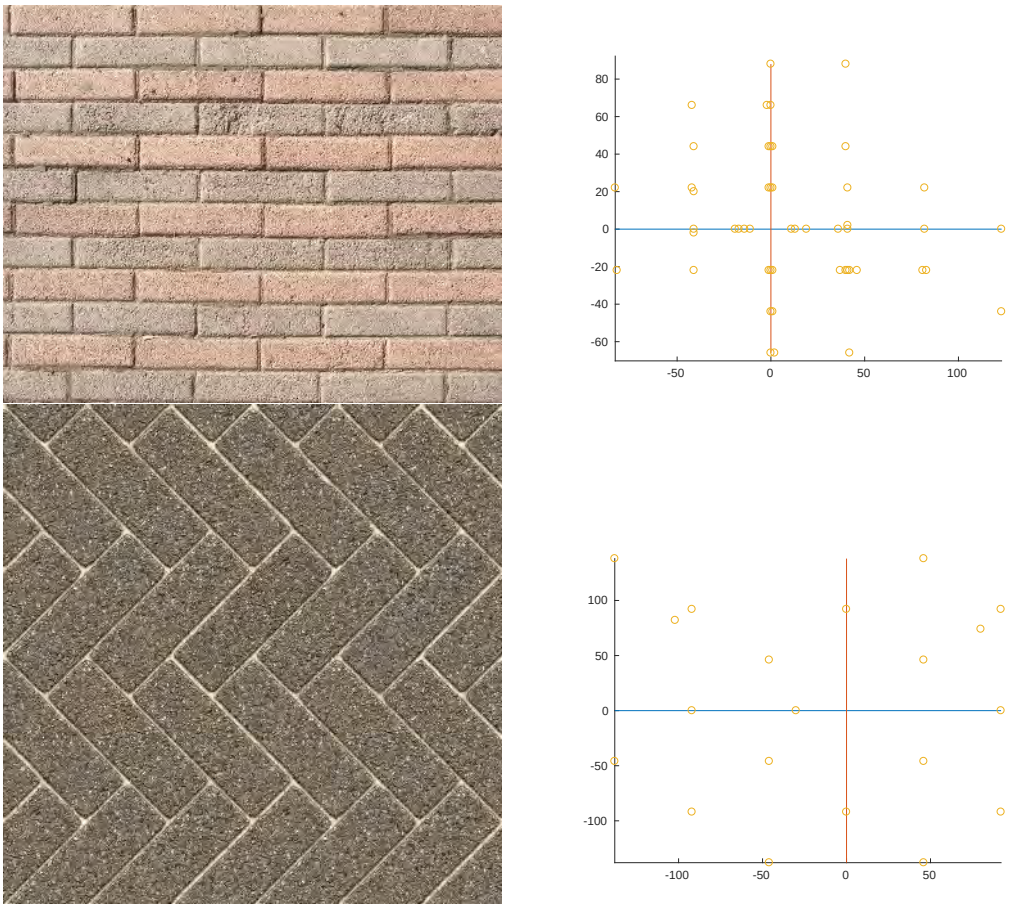


FIGURE 2.24 – Exemple d'*offsets* obtenus à partir d'une image. Pour chaque ligne, les points du graphe de droite représentent les *offsets* retenus dans l'image de gauche pour la suite de l'approche.

Algorithm 3 Algorithme Statistique+Graphcut

Entrée: Image I, masque M, ratio r
Sortie: Image complétée $I_{\text{inpainted}}$

Procédure STATISTIQUE+GRAPHCUT(I, M, r)
  $I_r \leftarrow$ RÉDUIRE(I, r)
  $M_r \leftarrow$ RÉDUIRE(M, r)
  $O \leftarrow$ CALCULEROFFSETS(I_r, M_r) ▷ Étape d'analyse
  $labels \leftarrow$ CALCULERLABELS(I_r, M_r, O)
  $labels \leftarrow$ AGGRANDIR$(labels)$
  $I_{\text{inpainted}} \leftarrow$ APPLIQUERLABELS$(labels)$
  Retourner $I_{\text{inpainted}}, labels$
Fin Procédure

Fonction CALCULERLABELS(I, M, O)
  $I_e \leftarrow I_{|\Omega \cup \partial\Omega}$
  Pour tout $p : (x, y) \in I_e$ **Faire**
    Si $p \in \partial M$ **Alors**
      $labels(p) \leftarrow 0$ ▷ Associer aux pixels adjacents le vecteur nul
    Sinon
      $labels(p) \leftarrow$ ALÉA$(1, |O|)$ ▷ Initialisation aléatoire
    Fin Si
    $labels \leftarrow$ GRAPHCUT(I_e, O, E_d, E_r)
  Fin Pour
  Retourner labels
Fin Fonction

Fonction $E_d(p, t, I, M)$
  Si $p \in \partial M$ **OR** $((p + t) \notin M)$ **Alors**
    Retourner 0
  Fin Si
  Retourner $+\infty$
Fin Fonction

où :

- E_d est le terme d'attache aux données pour chaque pixel p. Cette énergie pénalise les pixels qui choisissent les *offsets* qui translatent ces premiers dans Ω et non dans la zone connue $O \backslash \Omega$.
- E_r est l'énergie de régularisation pour chaque paire de pixels voisins (p_1, p_2). Cette énergie pénalise les pixels voisins qui ne choisissent pas le même *offset* afin de garantir une continuité au sein du masque Ω.

Remarque 2.4.3. La minimisation est faite sur $\overline{\Omega}_e$ et non sur Ω pour pénaliser les discontinuités entre Ω et le reste de l'image.

La minimisation est effectuée par une coupure de graphe [BK04 ; BVZ01] dont les terminaux sont les labels associés aux *offsets* et les nœuds sont les pixels de $\partial\Omega_e$ comme expliqué dans la section 2.3.3 et montré dans la figure 2.11. Il suffit enfin d'appliquer pour chaque pixel p l'*offset* associé au label l sélectionné durant la minimisation. La figure 2.25 montre un exemple de zone complétée après effacement du masque.

Nous constatons que dans le cas de textures avec un motif fortement structuré, l'approche fournit de très bon résultats là où l'approche basée sur *PatchMatch* fournissait

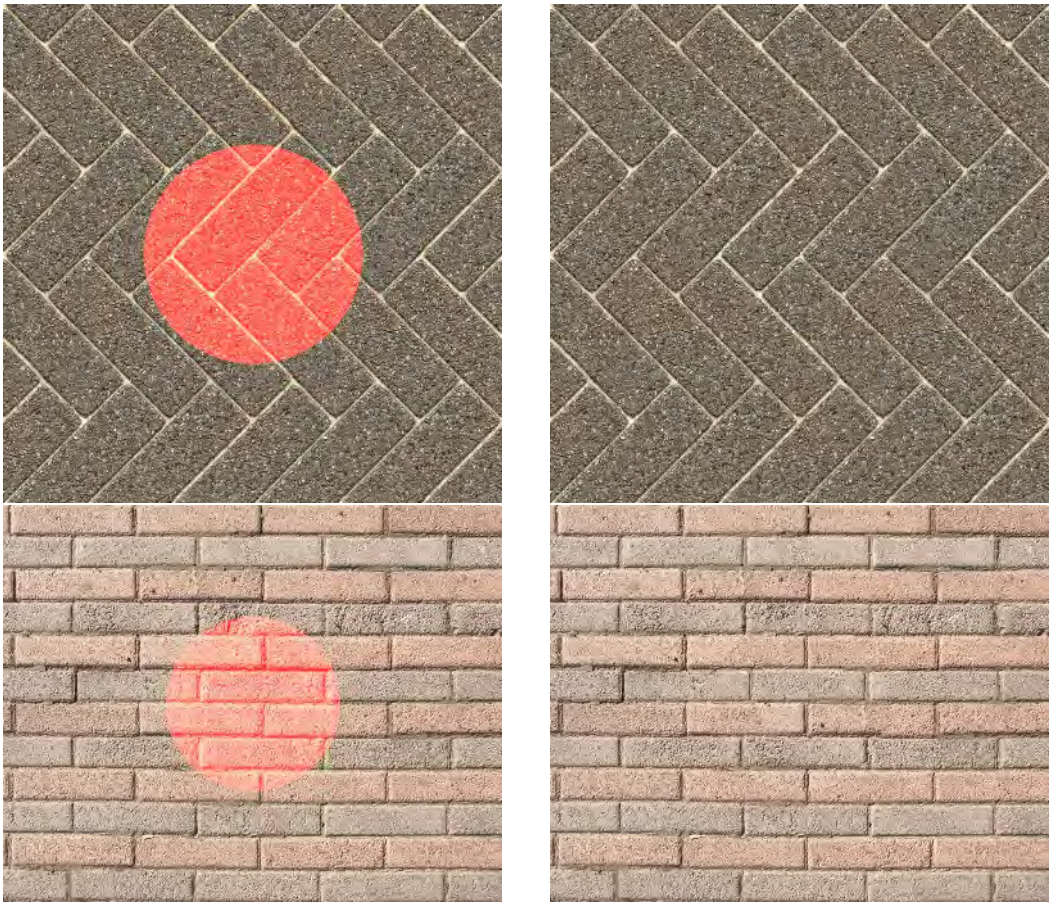


FIGURE 2.25 – Exemple d'*inpainting* en utilisant une approche par *offsets*. Gauche : image avec la zone masquée en surbrillance rouge. Droite : image après traitement.

des résultats moyens (par comparaison avec les résultats de la figure 2.22). Cela est dû au fait que seuls les *offsets* représentant la structure du motif sont sélectionnés pour minimiser l'énergie de l'équation 2.31. Le résultat est encore de qualité pour des textures à forte structure ayant de très légères déformations géométriques et colorimétriques. Cependant, dès que la texture possède des motifs qui sont moins structurés, le résultat obtenue est moins concluant (voir figure 2.26).

2.5 Les limites

Nous évoquons dans cette section les limitations d'un scénario de Réalité Diminuée basé essentiellement sur une approche d'*inpainting*. Chaque limitation sera détaillée et traitée dans un chapitre.

2.5.1 La restriction d'une seule approche d'*inpainting*

Deux points sont à noter. Premièrement, nous pouvons constater que les deux approches détaillées dans la section 2.4 fonctionne convenablement que selon un type de texture particulière. D'un côté nous avons l'approche d'*inpainting* basé sur *PatchMatch* qui fonctionne sur des textures ou images avec peu de structure. Cependant, les textures avec des motifs à forte structure seront difficilement complétées avec celle-ci car l'information sur ces structures n'est pas prise en compte durant l'effacement.

FIGURE 2.26 – Limites de l'approche par analyse statistique : cas d'une texture avec un motif non structuré. Gauche : vérité terrain. Milieu : résultat avec l'approche par analyse statistique. Droite : résultat avec l'approche par *PatchMatch*. Nous constatons pas mal de discontinuités dans l'image du milieu.

À l'inverse, l'approche basée sur les *offsets* fonctionne bien sur des textures avec des motifs fortement structurés. Cependant, elle aura du mal à compléter des macro-textures avec des motifs très peu structurés.
Or, dans notre contexte de scène d'intérieur, nous devons compléter généralement les textures présentes sur le sol et sur le mur. Ces textures peuvent avoir un motif fortement structuré et qui se répètent, comme par exemple du carrelage ou des briques. À l'inverse, la texture peut avoir un motif qui est peu structuré (du bois par exemple) ou ne pas avoir de motif du tout (du lino par exemple).
Il faut donc pouvoir choisir quelle méthode choisir pour compléter au mieux la texture d'une scène d'intérieur.

De plus, on peut avoir des textures comme celles de la figure 2.27 où aucune des deux approches ne fonctionnent convenablement ; nous proposons de créer une nouvelle approche, hybride, pour ces textures. Cette approche est décrite dans le chapitre 3.

2.5.2 La variation de luminosité

Dans un environnement d'intérieur, il est fréquent d'avoir une ou plusieurs sources lumineuses. De ce fait, les textures présentes sur les murs sont assujetties à des variations d'intensité lumineuse. Or les approches d'*inpainting* classiques ne tiennent pas compte de ces variations. Cela peut conduire à des artefacts de luminosité comme l'illustre la figure 2.28. Ceux-ci ont deux origines :

— la propagation non contrôlée de la texture faiblement éclairée dans une zone où on attendrait une texture fortement éclairée ;
— l'absence de texture soumise à une intensité moyenne au centre de l'image. On ne peut donc compléter qu'avec de la texture trop faiblement ou trop fortement éclairée.

FIGURE 2.27 – Cas d'une texture qui n'est complétée de manière optimale ni par *PatchMatch* (gauche), ni par les *offsets* (droite).

Dans le pipeline de SILTANEN [Sil15], une première gestion de ces variations est effectuée. Elle consiste à propager la variation de luminosité via des points de contrôle situés autour du masque Ω. Cependant, dans des situations comme celle montrée sur la figure 2.31, on constate que la propagation perd de son efficacité quand on arrive au centre du masque.

Nous proposons de mieux contrôler la propagation de la luminosité en complétant les isophotes traversant les masques, souvent de grande taille. Cette méthode est décrite dans le chapitre 5.

2.5.3 La variation de résolution

Dans le cas d'un scénario de Réalité Diminuée, nous sommes amenés à effectuer des homographies pour éliminer la déformation perspective des plans dans une image. Plusieurs approches proposées utilisent ces transformations pour obtenir de meilleurs résultats.

HUANG *et al.* [Hua+14] améliore également l'approche de HE et SUN [HS12b] en ajoutant une première gestion de la perspective. Pour cela, ils devinent la présence de plan dans l'image en recherchant, via les lignes de fuite présentes dans l'image, les points de fuite associés. Ils attribuent ensuite pour chaque pixel une probabilité selon laquelle ils appartiennent à un plan détecté. Ensuite, ils calculent, pour chaque plan, une pure transformation de perspective. Ils effectuent ensuite la recherche de décalage

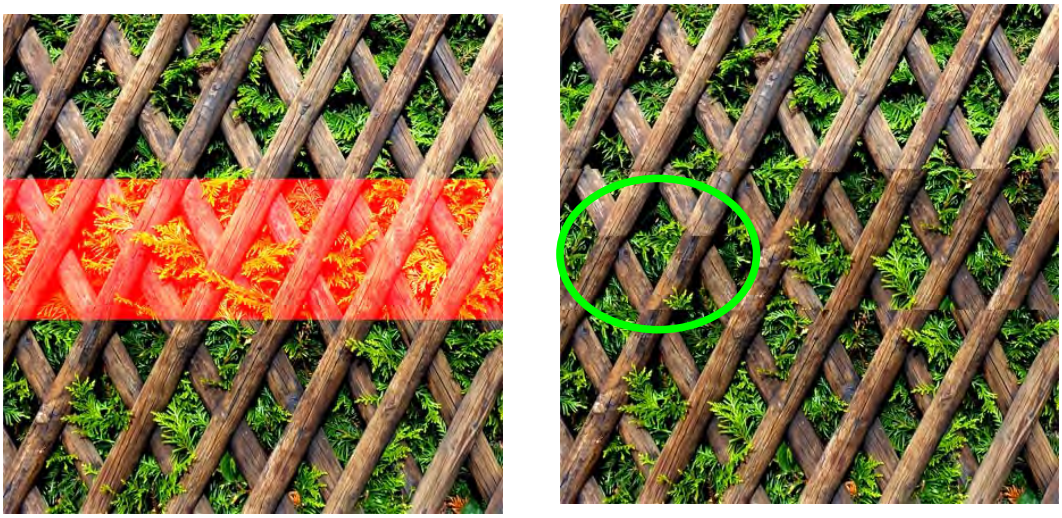


FIGURE 2.28 – Problème de respect de la luminosité. (image de gauche) Le masque à compléter se situe entre une zone plus « claire » et une zone plus « sombre ». (image de droite) L'*inpainting* conduit à des artefacts de luminosité, ici, des discontinuités.

dans l'espace rectifiée. Comme ils font la recherche dans l'image d'origine, ils modifient également l'énergie pour la diviser en deux : une énergie, dite de cohérence, pour évaluer la ressemblance entre un *patch* du masque et un *patch* de la zone connue dans l'espace rectifiée et une énergie, dite de guidage, pour forcer à associer les pixels qui ont une probabilité proche d'appartenir à un même plan ainsi ceux qui sont proches.

Si les résultats sont très convaincants dans le cadre de l'effacement d'un élément dans une seule image, elle reste limitée car elle dépend essentiellement de la recherche des points de fuite. De plus, il n'y pas de modèle 3D pris en considération ce qui limite l'utilisation pour un cadre de scénario de Réalité Diminuée. Notons enfin qu'il peut avoir des problèmes de segmentation à la frontière entre deux plans car la probabilité qu'un pixel appartient à un des deux plans est identique. Il est donc préférable d'utiliser le principe de rectification utilisé par KAWAI *et al.* [KSY15] et SILTANEN [Sil15] pour effectuer l'*inpainting* tout en gérant la perspective et la segmentation.

Cependant, des données ayant la même taille dans l'espace 3D n'ont pas la même résolution dans l'image selon qu'elles soient proches ou éloignées. Après rectification, les données les plus éloignées seront plus éparses dans l'image rectifiée que les données les plus proches. Si cette différence de résolution n'est pas prise en compte, les données éparses risquent d'être copiées dans une zone où l'on devrait générer une donnée de meilleure qualité. Cela crée un rendu flou dans cette zone une fois la reprojection dans l'image d'origine effectuée, comme illustré par la figure 2.31.

Dans cette thèse nous proposons une solution basée sur un critère de confiance. Cette solution est expliquée dans le chapitre 4.

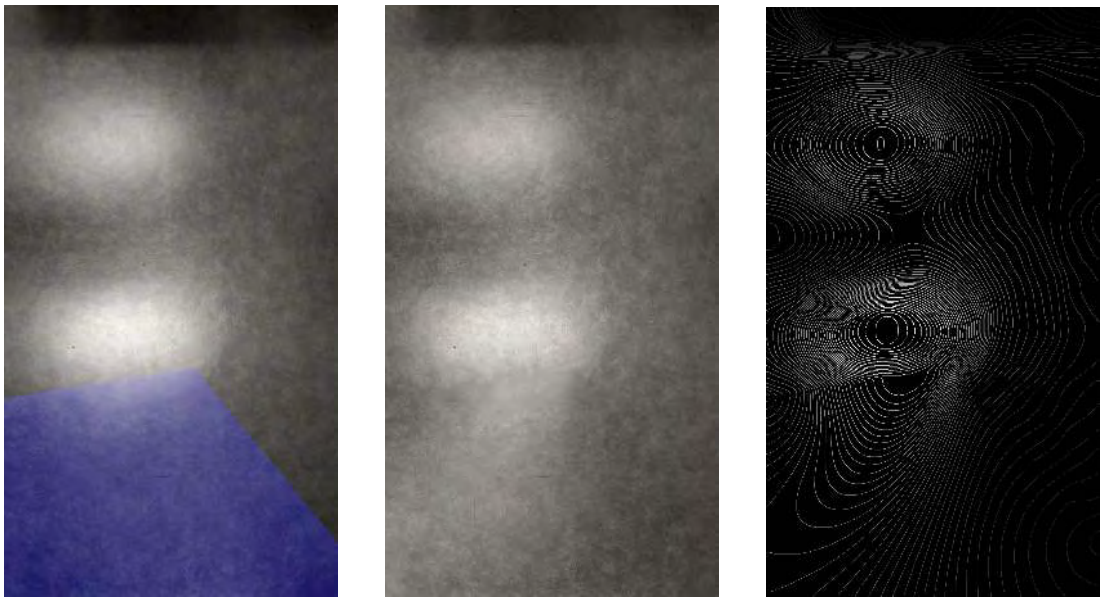


FIGURE 2.29 – Problème de respect de la luminosité. Dans cette configuration, la propagation de la luminosité dans le masque (zone en bleu de l'image de gauche) par les points de contrôle [Sil15] n'est plus optimale au centre du masque (image du milieu) car trop éloigné d'une information de luminosité. Les lignes isophotes (image de droite) ne sont plus respectées dans le masque.

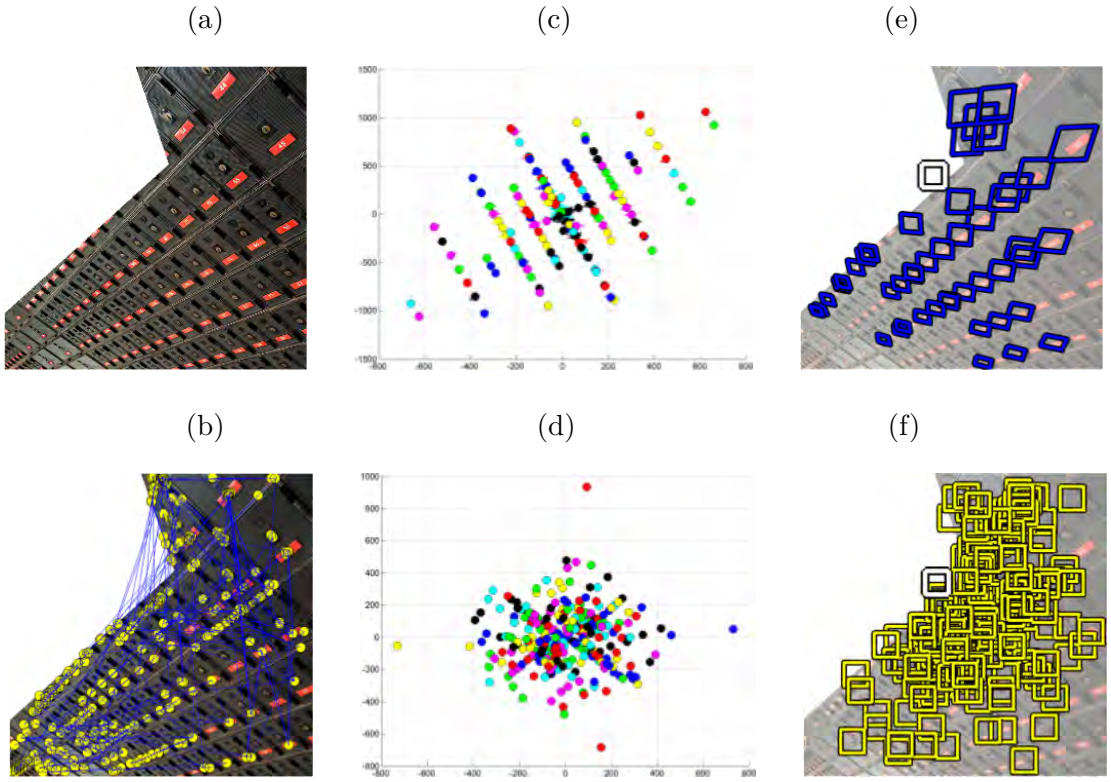


FIGURE 2.30 – Approche d'*inpainting* en tenant compte de la perspective (images extraites de [Hua+14]). Après le calcul des points d'intérêts (b) d'une image (a), les *offsets* sont calculés à partir de ces points d'intérêts. On constate une représentation des *offsets* qui respecte mieux la structure de la texture dans le cas de transformation de perspective (c) que dans le cas sans transformation (d). En conséquence, les *patchs* sélectionnés en bleu (e) comme candidats sont plus pertinents plus remplir le *patch* blanc que les *patchs* sélectionnés en jaune (f).

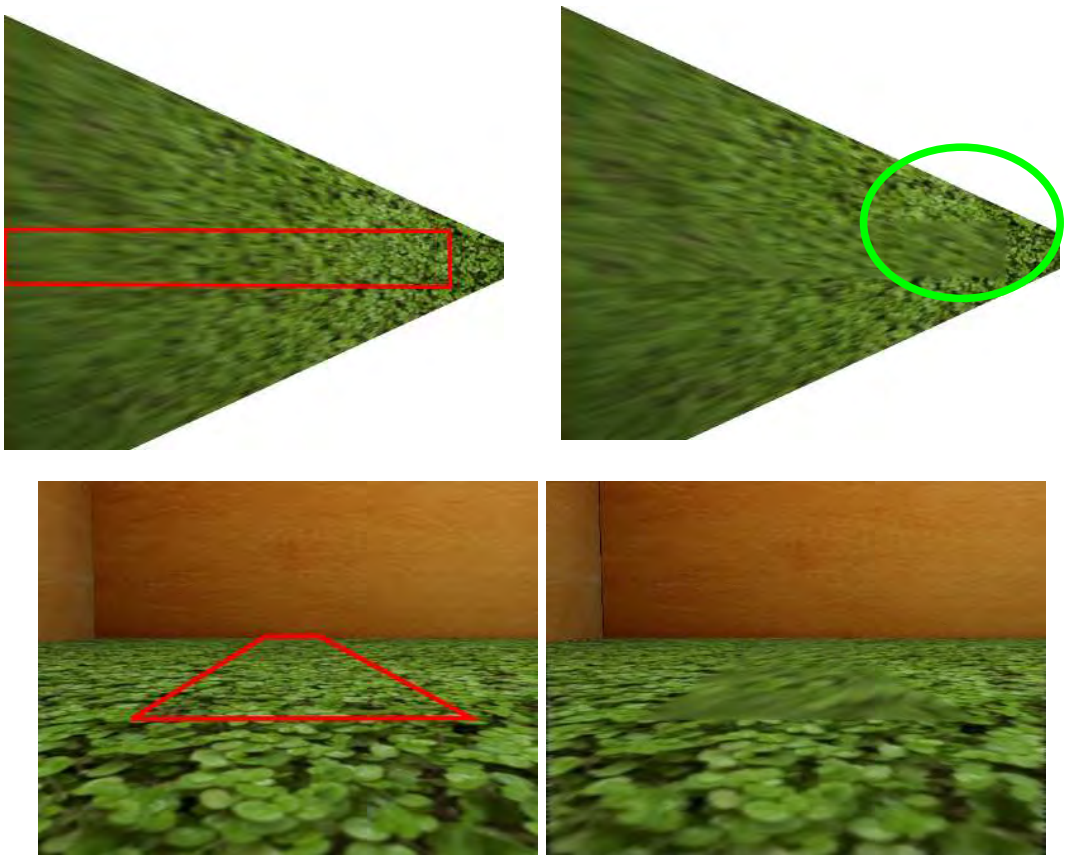


FIGURE 2.31 – Problème de la variation de résolution.

Deuxième partie

Contributions principales

Chapitre 3

Classification des textures et contraintes dans l'*inpainting*

Sommaire

3.1 Introduction

Dans le précédent chapitre, nous avons abordé le principe de l'*inpainting*. Nous avons également présenté les différentes catégories de méthodes d'*inpainting* ainsi que les approches les plus fréquemment utilisées. Nous avons en particulier mis en lumière les limites de chaque méthode (strictement en 2D) et notamment la qualité des résultats selon la structure de la texture.
Nous allons décrire dans ce chapitre les moyens de classer les textures selon leur structure et en conséquence selon la capacité à être complétées par une méthode d'*inpainting* spécifique. Cette analyse nous amène à développer une méthode de complétion originale donnant des résultats probants quelque soit le type de texture. Cette méthode se

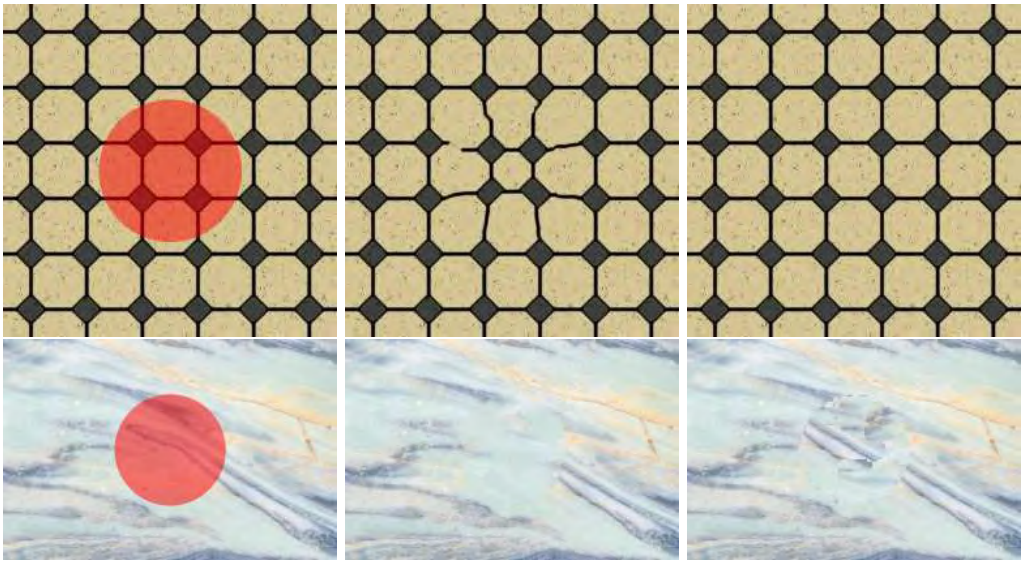


FIGURE 3.1 – Complétion de deux images (première colonne) selon deux méthodes différentes : *PatchMatch* (deuxième colonne) et approche statistique (troisième colonne)

caractérise par une initialisation de la zone masquée à partir d'une étude des *offsets* de l'image.

3.1.1 Les limites d'une seule approche d'*inpainting*

Nous avons vu plusieurs méthodes adaptées aux traitement des textures. Selon le type de texture, il est nécessaire d'appliquer une méthode spécifique pour obtenir des résultats optimaux. Dans la littérature, *PatchMatch* [Bar+09] est une approche fréquemment utilisée pour compléter des textures stochastiques et certaines textures semi-régulières. Cependant, les textures ayant une structure régulière ne sont pas complétées de manière fidèle car leur structure n'est pas prise en compte durant le processus de complétion.

À l'inverse, une approche statistique par coupure de graphe [HS12b ; Köp+15] sera optimale pour des textures régulières mais pêchera dès que le motif ne sera plus entièrement structuré. La figure 3.1 nous montre quelques cas limites des deux approches ci-dessus selon le type de texture. Sur la première ligne, la texture régulière est mieux complétée avec la deuxième approche. Sur la deuxième ligne, la texture stochastique semble plus cohérente avec la première approche.

Pour résoudre ces problèmes, nous proposons deux actions :

1. Pouvoir classer les textures en plusieurs catégories suivant leur structure. Cette classification permettra de choisir quelle algorithme d' *inpainting* utiliser. La classification se fait via une approche *a contrario*. Nous l'expliquons dans la section 3.2.

2. Construire une nouvelle méthode d'*inpainting* qui fonctionne indépendamment du type de texture ; pour cela, il faut gérer la présence ou non de structure majeure de la texture. Notre méthode se base sur une meilleure initialisation de la zone masquée. La section 3.3 donne les détails de notre approche. Des résultats sont présentés dans la section 3.4.

FIGURE 3.2 – Spectre des textures qui vont des textures régulières (gauche) aux textures stochastiques (droite) (image extraite de [Lin+06]).

3.2 Classification par approche *a contrario*

Dans cette section, nous abordons la problématique de la classification de textures en fonction de la méthode d'*inpainting* ayant la meilleure capacité à la compléter. Tout d'abord, nous rappelons les classifications de textures de l'état de l'art. Nous proposons ensuite une manière de classer les textures relatif à notre contexte de complétion.

3.2.1 Les types de textures

La classification de texture [TJ93] consiste à associer une image à une catégorie. Pour cela, chaque catégorie doit avoir des caractéristiques suffisamment discriminantes pour pouvoir dire si une texture appartient à cette catégorie. La difficulté est de déterminer ces différentes catégories.

Une classification connue consiste à introduire les textures *presque-régulières* dans le spectre des textures composées des textures régulières et des textures stochastiques [LL03]. Nous avons donc :

- Les textures **régulières** sont des textures caractérisées par un motif élémentaire qui est répété de manière identique à intervalles réguliers parfaits.
- Les textures **presque-régulières** (ou semi-régulières) sont les textures possédant une structure ayant subi une déformation géométrique ou colorimétrique.
- Les textures **irrégulières** sont les textures où il n'existe aucune motif régulier dans la répétition.
- Les textures **stochastiques** ou **aléatoires** sont les textures qui ont une structure aléatoire.

La figure 3.2 issue de [Lin+06] montre le spectre des textures segmenté par la partition définie ci-dessus.

Cependant, cette classification n'est pas suffisante car elle reste ambiguë. Une première ambiguïté concerne l'assimilation entre la couleur et la géométrie dans la définition d'une texture quasi-régulière.

LIU *et al.* [LLH04] proposent pour cela de distinguer la déformation géométrique de la déformation colorimétrique en créant deux axes d'évaluation : un pour la couleur et un pour la géométrie. La figure 3.3 montre des exemples de textures évaluées par ces deux critères.

Dans son monographe [Sau18], SAUVAGE étend cette classification à trois dimensions : la distribution du motif (uniforme ou non-uniforme, régulier ou irrégulier), la

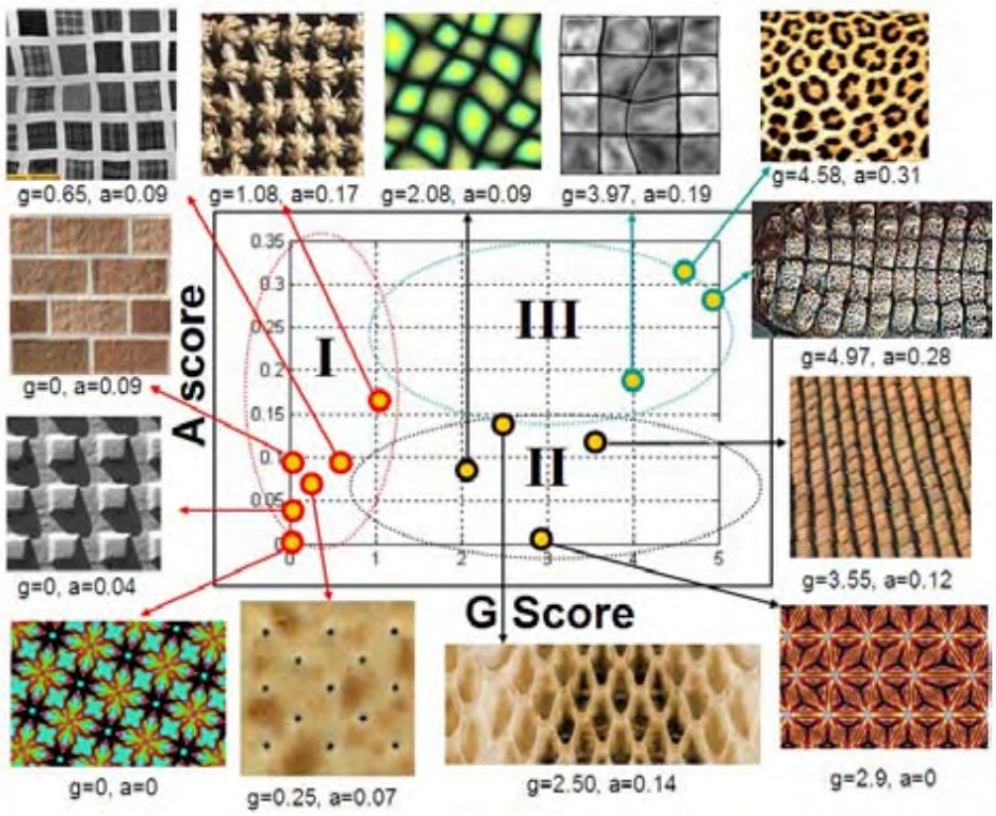


FIGURE 3.3 – Spectre des textures selon deux axes : un axe de déformation géométrique (axe des abscisses) et un axe de déformation colorimétrique (axe des ordonnées) (image extraite de [LLH04]).

variété des motifs (la répétition ainsi que des variations d'orientation, de formes et d'intensité) ainsi que le type de motif, selon qu'il soit structuré ou non-structuré.

Dans notre contexte de scénario de Réalité Diminuée de scènes d'intérieur, le spectre des textures étudiées est limité à des textures qui ont une distribution uniforme ainsi qu'un motif peu variable. En effet, la variation d'intensité (notamment due à la luminosité) est gérée dans le chapitre 5. Les variations d'orientation et de forme ne sont pas pris prises en compte dans ce chapitre non plus car nous traitons des textures à compléter sans variation de perspective (le traitement de la déformation perspective est l'objet du chapitre 4). Nous nous intéressons donc à deux dimensions : le type du motif et sa distribution.

Concernant la complétion d'une texture par une méthode d'*inpainting*, nous pouvons constater deux choses, d'après l'analyse faite dans le chapitre 2 :

— Les textures ayant une distribution régulière et un motif structuré sont bien mieux reconstruites avec une approche par analyse des *offsets* ; elles sont, à l'inverse, difficilement reconstruites avec une approche basée *PatchMatch*.
— Les textures ayant une distribution irrégulière ou ayant un motif non-structuré sont mieux reconstruites avec une approche par *PatchMatch* qu'avec une approche par *offsets*.

Nous pouvons dire que la classification de Liu *et al.* [LLH04] peut aider à déterminer quelle méthode d'*inpainting* choisir. Plus une texture est située proche de l'origine dans la figure 3.3, plus elle a de chances d'être mieux complétée par l'approche par *offsets*. À l'inverse, plus une texture s'éloigne de l'origine, plus elle a de chances d'être mieux complétée par *PatchMatch*. Le but de notre classification est d'associer une texture à la méthode d'*inpainting* qui complète au mieux une zone masquée de celle-ci. Pour cela, nous cherchons donc des directions symbolisant la structure et la distribution du motif. Cela nous amène à construire une méthode basée sur une approche *a contrario*. L'intérêt de l'approche *a contrario* est de détecter des éléments structurels dans une image. Dans notre cas, les éléments de structure sont les directions caractérisées par les *offsets*. Le modèle prédéfini est un modèle définissant une texture stochastique. Nous détaillons notre approche de détection par une méthode *a contrario* dans la prochaine sous-section.

3.2.2 Principe

Remarque 3.2.1. L'objectif de cette section est de résumer de manière succincte le principe de l'approche *a contrario* en définissant les paramètres principaux et en proposant un exemple simple pour comprendre le procédé.

Le principe *a contrario* a été introduit pour la première fois en traitement de l'image par Desolneux *et al.* [DMM00]. Il est couramment utilisé dans la détection de primitives géométriques comme des lignes [Gro14] et ou des ellipses [PGG17], mais aussi pour des structures de plus haut niveau comme des visages [LRP17] ou pour la détection de changements à partir de séquences d'images satellite [RMH10].

Dans le contexte des images, le principe *a contrario* stipule qu'il n'est pas possible d'exclure la détection d'éléments sur une image stochastique.

Il existera toujours (même rarement) des éléments de structure dans une image. L'idée est donc de mettre en place plutôt un seuil de détection basé sur le nombre moyen de détections que l'on aurait dans une donnée non structurée, et ainsi considérer ces données comme de faux positifs. Il faut que ce seuil ne soit pas trop bas, sinon des détections accidentelles seront comptabilisées ce qui nuit à la classification. À l'inverse,

ce seuil ne doit pas être trop élevé. En effet, les détections accidentelles, mais aussi les détections des structures, seront refusées

Nous notons H_0 le modèle stochastique générant donc des données non structurées par nature. Nous appliquons une méthode de détection à une donnée générée par ce modèle. Nous notons $\mathcal{E}_H$ l'ensemble des événements de détection d'éléments dans H_0. L'esprit des méthodes *a contrario* est de lier le nombre de détections dans une donnée H_0 au seuil de détection.

Définition 3.2.1. Les événements de $\mathcal{E}_H$ sont dits ϵ-significatifs dans H_0 si $|\mathcal{E}_H| \leqslant \epsilon$. C'est à dire que l'espérance du nombre d'événements détectés dans une donnée générée par H_0 est inférieure à ϵ.

Plus ϵ sera petit, plus petit sera le nombre de faux-positifs acceptés et plus strictes seront les conditions imposées pour détecter la structure. Considérons une donnée x quelconque (donc non générée par le modèle H_0). Par exemple, dans notre cas, cela peut être une image d'une texture quelconque. Nous définissons un ensemble $\mathcal{C} = \{c_1, ..., c_{N_C}\}$ de N_C de tests à appliquer sur la donnée x. Nous définissons également une fonction $k(c_i, x)$ qui mesure le nombre d'occurrence de la structure recherchée dans la donnée x avec le test c_i.

Exemple 3.2.1. Pour la détection de droites, on associe à une image sa carte de directions (calculée via le gradient de l'image). Un test est effectué sur chaque pixels, donc l'ensemble des tests est l'ensemble des pixels de l'image.

Une droite est détectée si un ensemble de pixels voisins partagent la même direction. Dans ce cas, H_0 correspond à une distribution uniforme où chaque pixel a autant de chance d'avoir une direction. On considère $\mathcal{E}_H$ comme l'ensemble des pixels issus d'une image générée selon H_0 et qui est détecté d'après la fonction $k(c_i, x)$. Sans expliciter sa formulation, $k(c_i, x)$ va considérer un pixel $\mathbf{p}$ comme détecté si les n pixels où passe une "ligne", de longueur n partant de $\mathbf{p}$ dans la direction donnée par la carte de direction en $\mathbf{p}$, partagent la même direction (n est un paramètre donnant la longueur minimale pour considérer une "ligne" comme une droite).

Une autre quantité est définie dans l'approche *a contrario*. Il s'agit de la quantité $\mathrm{NFA}(c_i, x)$ (Nombre de Fausses Alarmes) associée au candidat c_i observé dans la donnée x. Elle est définie comme le produit entre le nombre de tests N_C et la probabilité d'observer des événements au moins aussi significatifs que dans le modèle stochastique H_0 :

$$\mathrm{NFA}(c_i, x) = N_C \cdot \mathbb{P}\left[k(c_i, X) \geqslant k(c_i, x)\right], \tag{3.1}$$

où X est une donnée qui est générée selon le modèle H_0.

Intuitivement, il s'agit du nombre de faux positifs, à savoir les candidats modélisés à partir du modèle stochastique (donc qui ne devraient pas être détectés) et qui sont détectés par la fonction k.

Remarque 3.2.2. Notons que plus la mesure $k(c_i, x)$ est élevée, plus la probabilité que $k(c_i, X) \geqslant k(c_i, x)$ sera faible. Par conséquence, plus la valeur NFA sera faible ; il y aura donc plus d'observations significatives. Cette dernière quantité a été introduite pour fournir un test et valider les événements de la manière suivante : une observation (c_i, x) est ϵ-significative si et seulement si

$$\mathrm{NFA}(c_i, x) \leqslant \epsilon. \tag{3.2}$$

Il est facile de vérifier cela vu que le test est équivalent à :

$$\mathbb{P}\left[k(c_i, X) \geqslant k(c_i, x)\right] \leqslant \frac{\epsilon}{N_C}. \tag{3.3}$$

Enfin, nous devons définir, pour un test c_i, un seuil de détection κ_i dans une donnée X modélisée par H_0 de sorte que l'observation (c_i, X) soit ϵ significative :

$$\kappa_i = \min \left\{ t \in \mathbb{R}, \mathbb{P}\left[k(c_i, X) \geqslant t\right] \leqslant \frac{\epsilon}{N_C} \right\}, \tag{3.4}$$

où X est une donnée générée par H_0. La valeur κ_i correspond au plus petit seuil qui nous garantit que tout élément détecté au delà de ce seuil n'est pas un faux positif.

3.2.3 Un premier exemple

Pour comprendre l'approche *a contrario*, nous résumons l'exemple de la détection de la pression d'un bouton sur un objet détaillé dans [Gro14].

Considérons un premier exemple : nous observons une donnée binaire x. La valeur "0" (respectivement "1") correspond à l'absence (respectivement la présence) de pression sur le bouton. Nous voulons détecter la structure suivante : une suite suffisamment longue de "1" comme dans l'exemple ci-dessous :

$$0100100100011111111111111111001000010110011000110 0 \tag{3.5}$$

La structure peut être située à n'importe quel endroit de la donnée, chaque position doit donc être testée. L'ensemble $\mathcal{C}$ est défini comme l'ensemble des positions de x d'où peut partir la détection. Si x est de longueur L alors $N_C = L$.

Premièrement, une fonction d'observation $k(c_i, x)$ est définie. Elle permet de mesurer le degré auquel la structure recherchée est présente dans la donnée x chez le candidat c_i. Dans cette exemple $k(c_i, x)$ est la longueur de 1 consécutifs à la position c_i.
Ensuite, un modèle stochastique H_0 doit être défini pour la séquence binaire non structurée. Le choix le plus simple est le modèle du lancer de pièce : lancers indépendants avec probabilité égales de piles et de faces.
La prochaine étape consiste à définir le seuil κ_i. Pour cela il faut expliciter la probabilité $\mathbb{P}\left[k(c_i, X) \geqslant n\right]$ Dans cet exemple, il s'agit d'observer une suite d'au moins n "1" consécutifs commençant à la position c_i. En raison de l'indépendance du modèle stochastique, la probabilité est égale à celle où les n premiers binaires commençant à c_i sont tous égaux à 1 soit :

$$\mathbb{P}\left[k(c_i, X) \geqslant n\right] = p^n, \tag{3.6}$$

où p est la probabilité d'avoir 1 selon H_0.

Pour une séquence d'une longueur totale égale à 50, et avec $p = \frac{1}{2}$, le NFA d'une chaîne de longueur n vaut $50 \cdot 2^{-n}$. Deux cas de figures peuvent être considérées :

- Si on considère des séries de « un » élément, le NFA vaut 25. Il faut donc détecter 25 séries de « un » éléments avant de considérer la suivante comme significative ce qui n'est pas intéressant. En effet, il est absurde de déterminer l'existence d'une structure à partir d'un seul élément binaire.
  La conclusion est identique pour des séries de deux, trois ou quatre éléments. On peut dire que tant le NFA est au dessus de 1, les événements détectés sont assimilés au bruit, donc non identifiables.
- Si on considère des séries de six éléments, le NFA est en-dessous de "1". Cela signifie que la première détection peut être considérer comme significative.

3.2.4 Détection de la régularité d'une texture avec une méthode *a contrario*

Dans cette section, nous allons appliquer l'approche *a contrario* à la recherche d'*offset* dans une texture. Pour cela, nous explicitons le modèle auquel nous appliquons

l'approche *a contrario*.

Modélisation des *offsets*

Nous allons chercher, dans cette sous-section, à quantifier les directions dominantes d'une image à travers les *offsets*. Soit une image $I\colon U \to \mathbb{R}^n, p \mapsto I(p)$. Comme défini au chapitre 2, un *offset* $\overrightarrow{o}$ est un vecteur de translation qui à un pixel p associe $p + \overrightarrow{o}$ tel que ce dernier soit le pixel qui, au delà d'un seuil τ minimise une fonction de similarité entre deux patchs centrés en p et q :

$$\overrightarrow{o} = \text{argmin}_{\overrightarrow{u}\in\mathbb{Z}^2,|\overrightarrow{u}|>\tau} \left(\text{sim}\left(p, p + \overrightarrow{u}\right)\right), \tag{3.7}$$

où $\text{sim}(p, q)$ est une fonction de similarité entre les patchs centrés en p et q ; τ est un seuil pour éviter de considérer la solution évidente $(0, 0)$.

Nous considérons $\mathcal{O} \subset \mathbb{Z}^2$ l'ensemble des *offsets* pour I :

$$\overrightarrow{o} \in \mathcal{O} \Longleftrightarrow \exists p \in U, \overrightarrow{o} = \text{argmin}_{\overrightarrow{o}\in\mathbb{Z}^2,|\overrightarrow{o}|>\tau} \left(\text{sim}(p, p + \overrightarrow{o})\right). \tag{3.8}$$

Nous notons s la fonction qui à tout pixel p associe son offset $\overrightarrow{o}$:

$$s\colon U \to \mathcal{O}, p \mapsto s(p) = \overrightarrow{o}. \tag{3.9}$$

Chaque *offset* $\overrightarrow{o}$ possède une occurrence occurence$(\overrightarrow{o})$ définie de la manière suivante :

$$\text{occurence}(\overrightarrow{o}) = \sum_{p\in U} \mathbb{1}\left(s(p) = \overrightarrow{o}\right). \tag{3.10}$$

Pour un *offset* $\overrightarrow{o} = (u, v) \in \mathcal{O}$, on dit qu'il appartient à la droite vectorielle $\delta = Vect(u_\delta, v_\delta)$ si

$$\left|\det \begin{pmatrix} u & u_\delta \\ v & v_\delta \end{pmatrix}\right| = 0. \tag{3.11}$$

À une droite vectorielle δ on peut associer également une occurrence :

$$\text{occurence}(\delta) = \sum_{\overrightarrow{o}\in\delta} \left(\text{occurence}(\overrightarrow{o})\right). \tag{3.12}$$

Nous considérons D de cardinal N_D l'ensemble des droites vectorielles possédant au moins un *offset* ainsi qu'une relation d'ordre associée :

$$\forall(\delta_1, \delta_2) \in D, \delta_1 \leqslant \delta_2 \Leftrightarrow \text{occurence}(\delta_1) \leqslant \text{occurence}(\delta_2). \tag{3.13}$$

Nous notons $D_t = [\delta_1, \ldots, \delta_n]$ l'ensemble D trié selon la relation d'ordre ci-dessus.

Soit $\tau \in [0, 1]$, on définit pour une image I et ses droites D_t :

$$\text{premiers}(I, \tau) = \min_k \left(\sum_{i=1:k} occ(\delta_i) \; > \; \tau \sum_{i=1:N_D} occ(\delta_i) \right) \tag{3.14}$$

et

$$\text{population}(I, k) = \frac{\sum_{i=1:k} \text{occurence}(\delta_i)}{\sum_{i=1:N_D} \text{occurence}(\delta_i)}. \tag{3.15}$$

Cette valeur correspond aux k premières droites vectorielles nécessaires pour représenter le pourcentage τ de l'ensemble des droites vectorielles pondérées par leur occurrence.

Nous avons définies des notions dans cette sous-section qui quantifie les directions à travers les *offsets*. Nous allons, à partir de ces notions, construire un modèle de classification *a contrario* qui les intègrent.

Application de l'approche *a contrario*

Nous devons définir les paramètres suivants :
— l'ensemble $\mathcal{C}$ des tests ;
— la fonction d'observation k modélisant la structure de la texture ;
— le modèle H_0 pour les données stochastiques ;
— le nombre de faux positifs NFA ;
— la constante ϵ de ϵ-significativité de la définition 3.2.1.

Nous cherchons à déterminer les directions dominantes de la distribution du motif de la texture. Nous calculons donc les *offsets* présents dans l'image pour obtenir l'ensemble D_t des droites vectorielles défini à la sous-section 3.2.4.

À partir de cet ensemble, nous définissons les tests c_i et la fonction d'observation $k(I, c_i)$ pour le test c_i. Nous notons un test c_i avec $i \in [1, n] \subset \mathbb{N}$ comme l'ensemble constituée des n premières droites vectorielles de D_t.

La fonction d'observation est la fonction population définie par l'équation (3.15) et appliquée à (I, c_i) :

$$\text{population}(I, c_i) = \frac{\sum_{i=1:c_i} \text{occurence}(\delta_i)}{\sum_{i=1:N_D} \text{occurence}(\delta_i)}. \tag{3.16}$$

Quant au choix du modèle stochastique, il en existe plusieurs comme expliqués dans l'article de Materka et Strzelecki [M+98]. Il y a le modèle auto-régressif (AR) qui suppose une interaction locale entre les pixels de l'image où l'intensité du pixel est une somme pondérée des intensités des pixels voisins. Il y a ensuite le modèle des champs de Markov aléatoires (*Random Markov Fields (MRF)*) où la probabilité qu'un pixel soit dans un état défini est fonction des probabilités pour les états des pixels voisins. Il y a enfin le modèle fractal comme le mouvement brownien utilisé notamment pour modéliser les textures dont l'irrégularité est invariant selon l'échelle. Vu que l'on veut modéliser des textures non structurées, nous prenons une loi gaussienne centrée en une couleur μ avec une variance σ. Varier μ n'influence pas le modèle. σ est choisi de sorte à rester dans le domaine visible de l'image.
Concernant la détermination de la structure, nous devons déterminer $\mathbb{P}(k(c_i, X) \geqslant \tau)$ avec $\tau \in \mathbb{R}$. Pour cela, nous partons de la distribution normale $\mathcal{N}(\mu, \sigma)$ de la couleur. Comme nos tests concernent les droites vectorielles de D_t, nous devons construire la distribution de ces droites. À partir de la distribution colorimétrique $\mathcal{N}(\mu, \sigma)$, nous pouvons construire une seconde distribution normale $\mathcal{N}'(\mu_i, \sigma_i)$ qui modélise distribution de la population représentée par les c_i premières droites vectorielles pour une texture modélisée par H_0. Il est facile de prouver, notamment avec un test de normalité, qui vérifie si une distribution est normal, que cette dernière distribution $\mathcal{N}'(\mu_i, \sigma_i)$ est normale. On a donc :

$$\mathbb{P}(k(c_i, X) \geqslant \tau) = 1 - \mathbb{P}(k(c_i, X) < \tau) = 1 - F_X(\tau), \tag{3.17}$$

avec

$$F_X(x) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x - \mu}{\sqrt{2}\sigma_i}\right)\right), \tag{3.18}$$

où erf est la fonction d'erreur :

$$\text{erf}\,(x) = \frac{2}{\sqrt{\pi}} \int_o^x e^{-t^2}\,\mathrm{d}\,t. \tag{3.19}$$

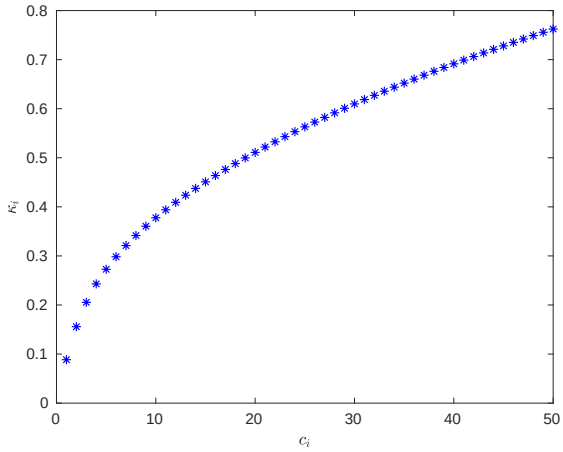


FIGURE 3.4 – Évolution du seuil κ_i en fonction du nombre de droites vectorielles c_i considérées.

Détermination des κ_i Nous déterminons donc pour chaque candidat c_i son seuil κ_i. Pour rappel (*cf.* équation 3.4),

$$\kappa_i = \min \left\{ \tau \in \mathbb{R}, \mathbb{P}(k(c_i, X) \geqslant \tau) \leqslant \frac{\epsilon}{N_C} \right\}. \tag{3.20}$$

Ce qui donne, en appliquant les équations (3.17) et (3.18) à l'équation 3.20 :

$$\kappa_i = \min \left\{ \tau \in \mathbb{R}, 1 - \frac{1}{2} \left(1 + \operatorname{erf} \left(\frac{\tau - \mu}{\sqrt{2}\sigma_i} \right) \right) \leqslant \frac{\epsilon}{N_C} \right\}. \tag{3.21}$$

La figure 3.4 nous montre l'évolution de κ_i selon le candidat c_i.

Prise de décision

Maintenant que nous avons pour chaque test c_i son seuil κ_i associé, nous pouvons appliquer la fonction $k(c_i, x)$ à une image I représentant une texture T pour tous les tests c_i.
Nous nous retrouvons à la fin avec un vecteur V binaire de taille N_C. La valeur 1 est attribuée pour un test c_i si $f(c_i, x) > \kappa_i$, 0 sinon.

Nous avons effectué le processus sur chaque texture de deux *benchmarks* de texture de la littérature : VisTex [MIT02] et OuTex [Oja+02]. Nous calculons pour chaque texture la somme du vecteur binaire associé ; nous analysons la distribution de la somme calculée pour l'ensemble de textures. Une somme nulle ou quasi-nulle signifie que la texture a une forte chance d'être irrégulière. À l'inverse, une somme proche ou égale à N_C signifie que la texture a de fortes chances d'avoir un motif régulier. La figure 3.5 montre l'histogramme des deux *benchmarks* évoqués ci-dessus. Nous constatons un regroupement des textures aux extrémités de l'histogramme. Cela signifie que tous les tests effectués sont discriminants pour une grande majorité de textures. La classification par une approche *a contrario* est donc une approche fiable dans la mesure de la structure via les *offsets*. En effet, dans le cas d'une texture T, seuls quelques tests c_i peuvent fournir un résultat contraire aux autres tests de N_C. Seul une infime minorité de textures ont des résultats aux tests qui se contredisent comme ceux de la figure 3.6.
La distribution est donc bimodale où les modes sont situées aux bords de l'axe. Nous considérons donc, pour la prise de décision, qu'une texture est suffisamment régulière

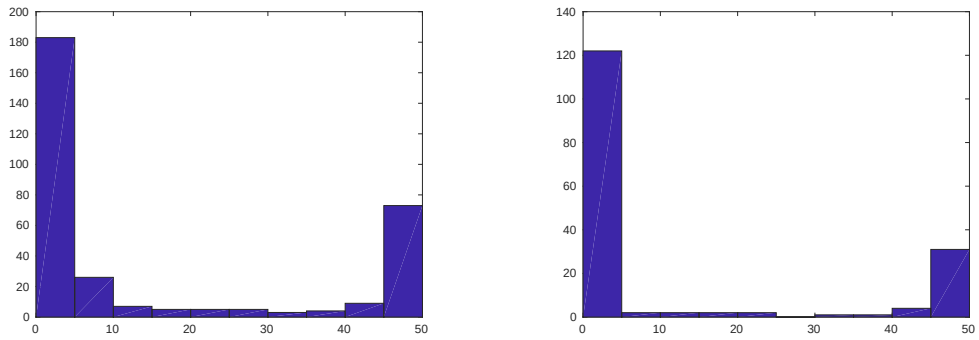


FIGURE 3.5 – Histogramme de distribution des *benchmarks* de données OuTex (gauche) et VisTex (droite). Axe des abscisses : nombre de tests validés par une texture. Axe des ordonnées : nombre de textures du *benchmark* validant n tests (n indiqué en abscisse). Nous constatons un regroupement des textures aux extrémités de l'histogramme. Cela signifie que tous les tests effectués sur une texture convergent vers la même décision pour une grande majorité de textures. Si les *offsets* appartiennent à peu de directions, il n'y aura que beaucoup de directions dominantes. Donc la quasi-totalité des tests rendront un résultat positif. Cette texture sera donc assimilée à une texture ayant un motif structuré.

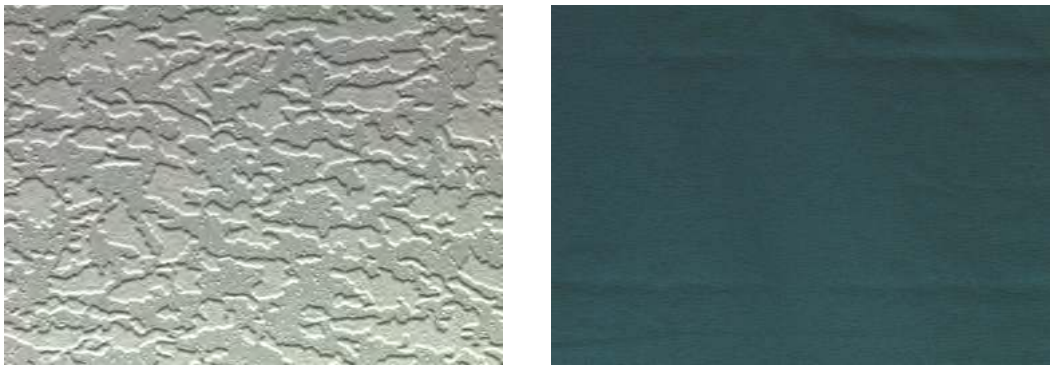


FIGURE 3.6 – Exemples de textures dont les résultats aux tests se contredisent. Dans les deux cas on peut constater effectivement une structure horizontale sur les deux textures mais que ne doit être suffisamment marquée pour être détectée à chaque fois.

pour être complétée par une approche par *offsets* si et seulement si elle valide le test final suivant :

$$\sum_{i}^{N_C} V(i) > \frac{N_C}{2}. \tag{3.22}$$

3.2.5 Résultats

Dans cette section, nous testons cette classification sur deux ensembles de textures : l'un construit par nous-même et deux autres de la littérature. Nous considérons aussi un ensemble de textures créé aléatoirement avec une couleur moyenne $\mu_c = [200, 100, 100]^T$ et un écart-type $\sigma_c = 20$ de 500 images stochastiques de taille 640×480 pour générer la distribution des droites vectorielles selon H_0. Un exemple d'image générée est donnée dans la figure 3.7.

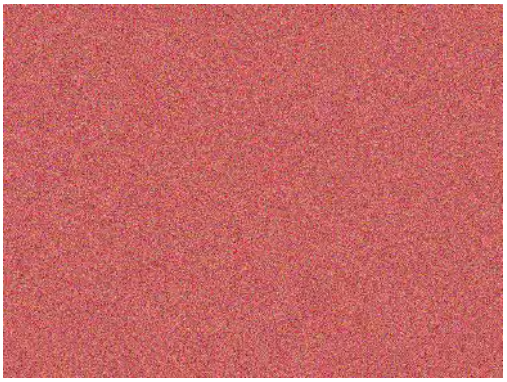


FIGURE 3.7 – Exemple d'image générée par le modèle $H_0 = (\mu_c, \sigma_c)$ avec $\mu_c = [200, 100, 100]^T$ et $\sigma_c = 20$.

FIGURE 3.8 – Exemples de texture du set personnalisé. Première ligne : texture ayant le label `OFFSETS`. Deuxième ligne : texture ayant le label `PATCHMATCH`.

Benchmark personnalisé

Ce *benchmark* de données est composé de deux labels : `PATCHMATCH` si la texture est complétée au mieux avec *PatchMatch* et `OFFSETS` si la texture est complétée au mieux avec une approche par *offsets*. L'attribution, pour une texture, à un label a été fait en effectuant une complétion, par les deux approches, dans un masque circulaire situé au centre de l'image. Le choix a été fait en analysant de manière qualitative le résultat à l'intérieur du masque. La figure 3.8 montre quelques exemples des deux catégories de ce *benchmark*.

Le tableau 3.1 montre le taux de réussite de l'approche *a contrario* pour chaque label.

benchmark	`PATCHMATCH`	`OFFSETS`	précision
personnel	75.00%	87.36%	81.23%
VisTex	92.24%	54.90%	80.84%
OuTex	81.93%	67.62%	81.93%

TABLE 3.1 – Résultat de la classification *a contrario* sur le *benchmark* personnalisé et les *benchmarks* VisTex [MIT02] et OuTex [Oja+02].

Méthode	Auteur	Approche	Technique	*benchmark*	Précision (%)
1	Tou *et al.* [TTL09]	matrice de co-occurence	GLCM (12 carac.)	N/R	85.73
			GLCM (10 carac.)	N/R	78.15
2	Tuzel *et al.* [TPM06]	matrice de covariance	contours	N/R	84.65
			GLCM	N/R	79.94
			filtre de Gabor	N/R	91.86
3	Tou *et al.* [TTL09]	Filtres	Gabor (12 filtres)	autre	89.74
			Gabor (24 filtres)	N/R	91.86
4	Zhao *et al.* [Zha+11]	structure et statistique	LBP	Broadatz	79.97
				UIUC	57.59
			CLBP	Broadatz	86.63
				UIUC	83.29
5	Tou *et al.* [TTL09]	combinaisons	GLCM et Gabor	N/R	91.06
6	proposé	*a contrario*	*offsets*	OuTex	81.93

Table 3.2 – Comparaison avec d'autres approches (tableau issu de Patil [Pat13]). N/R : information non renseignée concernant le *benchmark* utilisé dans l'article d'origine.

Benchmarks de la littérature

Les deux autres *benchmarks* utilisés sont VisTex [MIT02] et OuTex [Oja+02]. OuTex possède 320 images représentant des textures très diverses (sables, tapis, sol, bois). VisTex possède 167 images représentant des types de textures diverses (sables, feuilles, carrelage, briques, bois). Nous nous limitons uniquement aux images représentant une texture d'intérieur sans perspective. Ce qui fait donc environ 150 images pour VisTex et 250 images pour OuTex. Les labels de ces deux *benchmarks* étant nombreux, nous associons les textures à chaque label, soit au label `PATCHMATCH` (par exemple les labels `barleyrice,crushedstone,flour`), soit au label `OFFSETS`, (par exemple les labels `canvas,carpet,paper,wallpaper`). Pour calculer notre taux de précision et comparer avec les autres méthodes, nous regroupons les labels des *benchmarks* VisTex et OuTex en deux catégories : l'une où les textures sont globalement mieux reconstruites avec *PatchMatch* et l'autre où les textures sont globalement mieux reconstruites avec les *offsets*. Certains labels (par exemple le label `tapis`) contiennent des textures pour lesquelles l'approche *PatchMatch* sera plus optimale alors que d'autre textures du même label seront mieux reconstruites avec les *offsets*.

Analyse

Nous constatons que le taux de réussite est globalement bon sans être exceptionnel. Notons qu'il est supérieur pour les textures se complétant mieux avec les *offsets* que les textures se complétant avec *PatchMatch*. Il est préférable d'avoir ce meilleur taux pour le label `OFFSETS` car il est plus pénalisant de compléter une texture à forte structure avec *PatchMatch* que compléter une texture irrégulière avec les *offsets*.
Nous comparons également notre approche avec d'autres méthodes de la littérature. Le tableau 3.2 résume l'ensemble des méthodes avec leurs taux de précision. Il est difficile de faire une comparaison dans la mesure où notre objectif n'est pas de classer par exemple une texture comme du bois ou de la brique mais de classer en fonction de la méthode d'*inpainting* la plus adaptée.

Globalement nous constatons que malgré cette différence entre les labels des deux *benchmarks* et notre manière de classer, nous obtenons des résultats convenables et assez proches des taux de précision, voire mieux si on compare par exemple à Zhao *et al.* [Zha+11].

3.2.6 Limites

Si cette approche donne des résultats corrects, elle a cependant des limites qui méritent des approfondissements. Premièrement, elle dépend beaucoup du modèle stochastique qui peut ne pas être le plus représentatif d'une texture irrégulière. Deuxièmement, elle ne garantit pas la meilleure méthode de complétion possible pour une texture entre *PatchMatch* et les *offsets* si celle-ci ne peut être reconstruite correctement par aucune deux méthodes utilisée (comme par exemple, la texture de la figure 3.9). Pour reprendre cette approche dans le cadre du scénario de diminution, nous limiterons son utilisation uniquement pour décider si l'on peut compléter une texture par une approche de synthèse (création d'une nouvelle image) ou non. Dans la suite de ce chapitre, nous proposons une approche de complétion capable de gérer les texture qui ne sont pas correctement complétées par l'une des deux méthode, comme celle de la figure 3.9.

FIGURE 3.9 – Cas d'une texture qui n'est complétée de manière optimale ni par *PatchMatch*, ni par une approche par *offsets*.

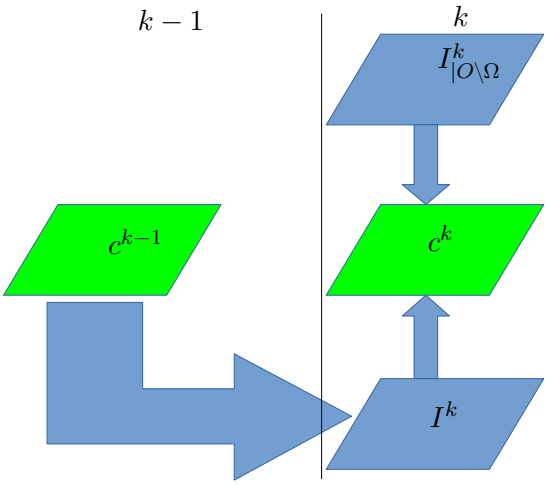


FIGURE 3.10 – Schéma de la mise à jour de la carte de correspondances

3.3 Méthode d'*inpainting* avec contraintes de structure

Face au problème de complétion des textures quasi-régulières, nous présentons une méthode d'*inpainting* capable de fournir un rendu optimal quelque soit le type de texture. Pour cela, nous allons combiner l'analyse statistique des *offsets* à la méthode de complétion par l'algorithme *PatchMatch*. L'idée est de créer une carte de correspondances à l'initialisation de *PatchMatch* qui tient compte des structures majeures d'une texture.

3.3.1 Le problème de l'initialisation

Contexte

Soit une image I définie sur un domaine $O \subset \mathbb{N}^2$ et un masque $\Omega \subset O$ des données à compléter. Dans *PatchMatch*, nous devons établir une correspondance entre les pixels appartenant à Ω et ceux appartenant à $O\setminus\Omega$. Pour la suite, nous notons c la fonction de correspondance :

$$c\colon O \to O\setminus\Omega \tag{3.23}$$
$$(x,y) \mapsto c(x,y) \tag{3.24}$$

où :

$$c(x,y) = \begin{cases} (x,y) & \text{si } (x,y) \in O\setminus\Omega \\ (u,v) \in O\setminus\Omega & \text{sinon.} \end{cases} \tag{3.25}$$

Comme énoncé dans le chapitre 2, *PatchMatch* est une approche multi-résolutions (ici, nous supposons que nous avons N résolutions). À chaque étage k de la pyramide de résolutions, la zone masquée de l'image $I^k_{|\Omega}$ est reconstruite à partir de la carte de correspondance c^{k+1} obtenue à l'étage supérieur $k+1$ (résolution plus faible) comme montré sur la figure 3.10. La carte de correspondances c^{k-1} est redimensionnée à la résolution de l'étage k avec une interpolation aux plus proches voisins et une multiplication par 2 de ses valeurs. Il suffit alors d'appliquer *PatchMatch* entre I^k sur laquelle on a appliqué une première fois c^k et $I^k_{|O\setminus\Omega}$.

Cependant, à l'étage N de plus faible résolution, nous devons créer la carte de correspondance c^N de I^N sans l'aide d'information d'un étage supérieur. Pour cela, nous

devons faire une première estimation de $I^N_{|\Omega}$ qui respecte la continuité à la frontière entre Ω et $O\backslash\Omega$ ainsi que la structure de la texture comme expliqué dans le chapitre 2.

Initialisation par diffusion

L'initialisation par diffusion consiste à appliquer une méthode d'*inpainting* par diffusion à $I_{|\Omega}$. Cette image initialisée I_{init} est ensuite redimensionnée jusqu'à atteindre l'étape N de la pyramide de résolution. Il suffit ensuite d'appliquer *PatchMatch* à I^N_{init} et à $I^N_{|O\backslash\Omega}$ pour créer la carte de correspondances c^N qui sera transférée ensuite vers les étages inférieurs.

Les méthodes utilisées sont celles présentées durant le chapitre 2 sur l'*inpainting* basée diffusion. Nous avons donc les méthodes suivantes :

— Méthode isotrope : la propagation s'effectue de la même manière dans toutes les directions comme l'équation de la chaleur : $\frac{\partial u}{\partial t} = k\nabla^2 u$ [Ber+00].
— Méthode de propagation selon les contours [SKC03].
— Méthode par propagation du flux [TD05].

Nous considérons trois textures : une texture stochastique, une texture presque régulière et une texture strictement régulière. La figure 3.11 présente les résultats de ces textures complétées.

Nous constatons que quelque soit la méthode de diffusion pour l'initialisation que :

— la texture stochastique est complétée de manière satisfaisante ;
— les textures régulière et presque régulière complétées n'ont pas préservé leur structure sur la partie complétée.

Cela s'explique par le fait que la diffusion va propager les valeurs des pixels voisins à Ω selon la direction des gradients des pixels de la frontière extérieure $\partial\Omega_e$. Cela garantie la continuité à la frontière (on ne voit pas d'artefacts entre Ω et $U\backslash\Omega$. Cependant, aucune information sur la structure globale n'est propagée dans le masque.

Initialisation par pelure d'oignons

Nous expliquons dans cette sous-section la méthode dite par pelure d'oignons ou *onion peel* [New+17]. Cette méthode complète une couche à la fois, remplissant progressivement la zone masquée, devenant l'initialisation pour la complétion au niveau N de résolution. Les premières couches ont une épaisseur d'un pixel et localisées à la frontière de Ω. Chaque pixel d'une couche est complété en utilisant l'information de la couche précédente déjà complétée. Les pixels d'une couche sont traité de manière parallèle. Si la couche à traiter est notée $\partial\Omega$, ils considèrent uniquement les pixels connus, noté Ψ'_p d'un *patch* Ψ_p centré en un pixel p :

$$\Psi'_p = \{q \in \Psi_p, q \notin \Omega\}\,. \tag{3.26}$$

Une fonction de similarité, notée $\sim$, est définie pour comparer les pixels d'un *patch* centré en un pixel de $\partial\Omega$. L'intensité du pixel p est définie comme une moyenne pondérée des valeurs des correspondants de ses voisins connus dans le *patch* Ψ_p :

$$I(p) = \frac{\sum_{q\in\Psi'_p} s(p,q)\; I(c(q))}{\sum_{q\in\Psi'_p} s(p,q)}, \tag{3.27}$$

où la fonction de pondération $s(p,q)$ accorde un poids plus important si sim est faible

$$s(p,q) = \exp\left(\frac{\mathrm{sim}(p,q)^2}{2\sigma^2}\right), \tag{3.28}$$

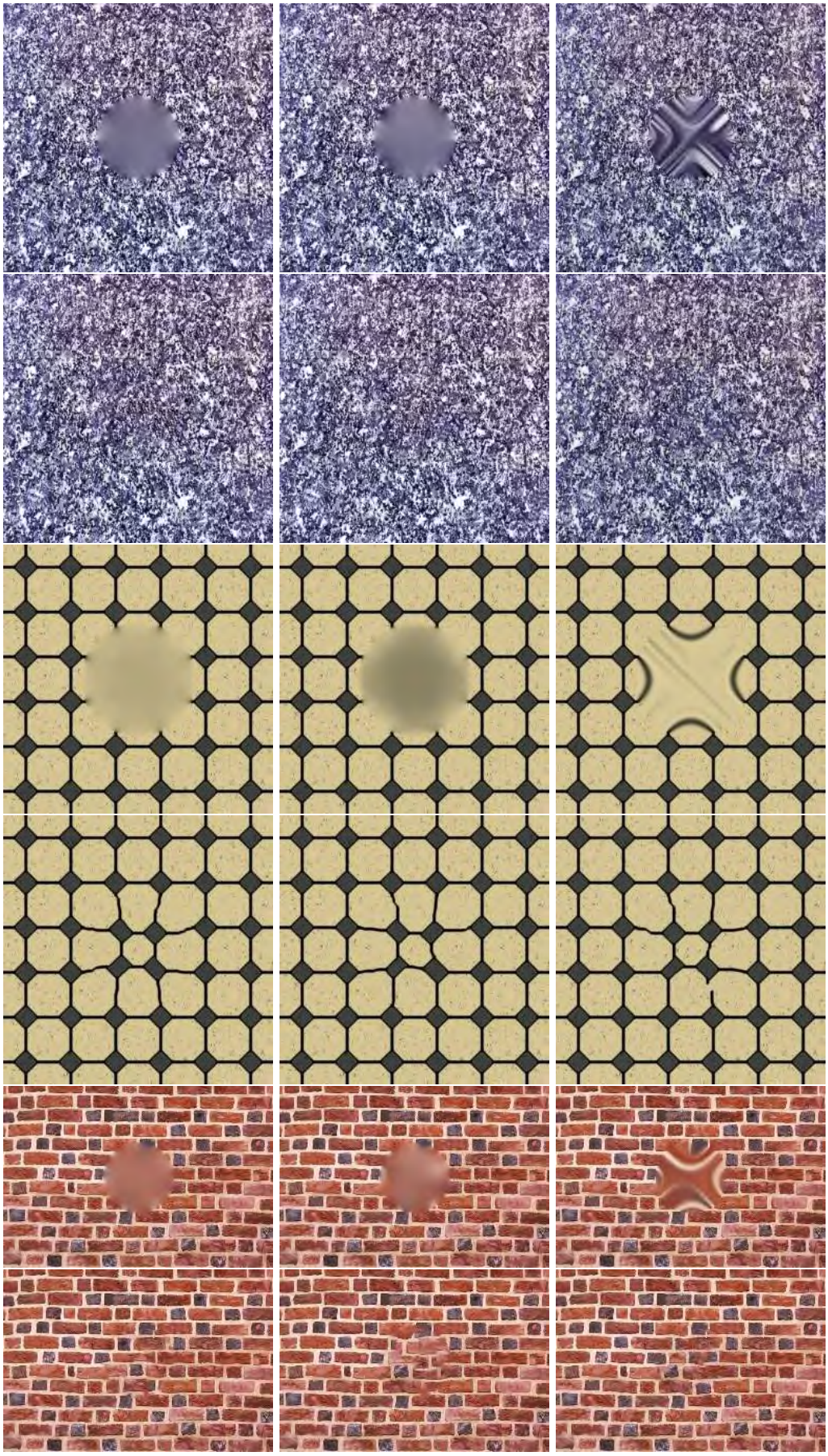


FIGURE 3.11 – Textures complétées par *PatchMatch* avec une initialisation par diffusion

FIGURE 3.12 – Comparaisons d'initialisation (figure issue de NEWSON *et al.* [New+17]). À part la première colonne qui montre la zone masquée (en vert) et la vérité terrain, les autres colonnes montrent l'initialisation par différentes techniques (première ligne) et le résultat final associé (deuxième ligne). De gauche à droite, les initialisations sont l'initialisation aléatoire, l'initialisation par diffusion et l'initialisation par pelure d'oignon. Nous constatons que seule la dernière initialisation donne un résultat convenable.

et c est la carte de correspondances initialisée de manière aléatoire.

La figure 3.12 présente différentes initialisations. On y constate que la méthode par pelure d'oignon propose une meilleure initialisation et un meilleur résultat que les autres initialisations. En effet, les deux première initialisations vont plutôt faire ressortir la couleur blanche du mur ce qui empêche *PatchMatch* de sélectionner des *patchs* venant du cigare. Notons que cette approche propage non seulement des valeurs de I mais aussi les valeurs d'une image $O \to \mathbb{R}^2$ appelée une carte de caractéristiques T définie comme un gradient normalisé :

$$T_x(p) = \frac{1}{\text{card}(\mathcal{N}_p)} \sum_{q \in \mathcal{N}_p} |I_x(p)|, \quad T_y(p) = \frac{1}{\text{card}(\mathcal{N}_p)} \sum_{q \in \mathcal{N}_p} |I_y(p)|. \tag{3.29}$$

L'ajout de cette nouvelle carte essaie donc de respecter la structure locale de la texture.

La figure 3.13 montre trois exemples de rendus finaux de NEWSON *et al.* par rapport à un *inpainting* par *PatchMatch* classique. Nous voyons bien que dans le cas des deux textures où il y a des motifs structurés, l'initialisation par pelure d'oignon arrive à propager la structure dans le masque ce qui n'est pas le cas de la diffusion.

Cependant, cette manière d'initialiser l'image pour la suite de la complétion ne fait que propager des valeurs directement voisines de Ω. On n'a aucune garantie d'un respect de la structure globale de la texture car celle-ci n'est pas intégré dans la manière d'initialiser I.

La figure 3.14 montre des cas de textures où l'initialisation n'a pas permis de fournir une texture complétée qui respecte la structure globale de la texture.

Nous avons vu que l'initialisation dans l'algorithme *PatchMatch* a une incidence sur la suite du résultat. Des initialisations, comme la diffusion, permet de compléter efficacement des textures ayant un motif non structurée. Cependant, dans le cas des textures ayant un motif structuré, une initialisation qui tient compte de cette structure doit être fournit. NEWSON *et al.* propose une initialisation basée sur la propagation par pelure d'oignon. Cette propagation est basée sur une approche d'*inpainting* gloutonne

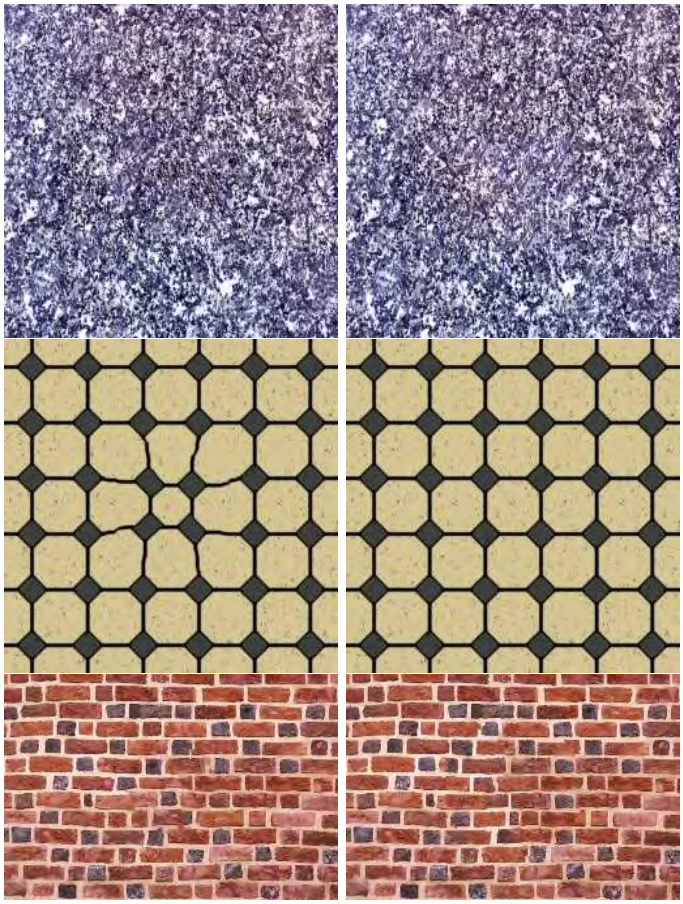


FIGURE 3.13 – Résultats finaux entre une initialisation par diffusion (colonne de gauche) et une approche par pelure d'oignons de NEWSON *et al.* [New+17] (colonne de droite). Si les résultats sont équivalents dans le cas de la première texture plutôt irrégulière (première ligne), ce n'est pas le cas pour la texture régulière (deuxième ligne) et la texture presque régulière (troisième ligne) où l'initialisation par pelure d'oignon donne des résultats finaux bien meilleurs que celle par diffusion.

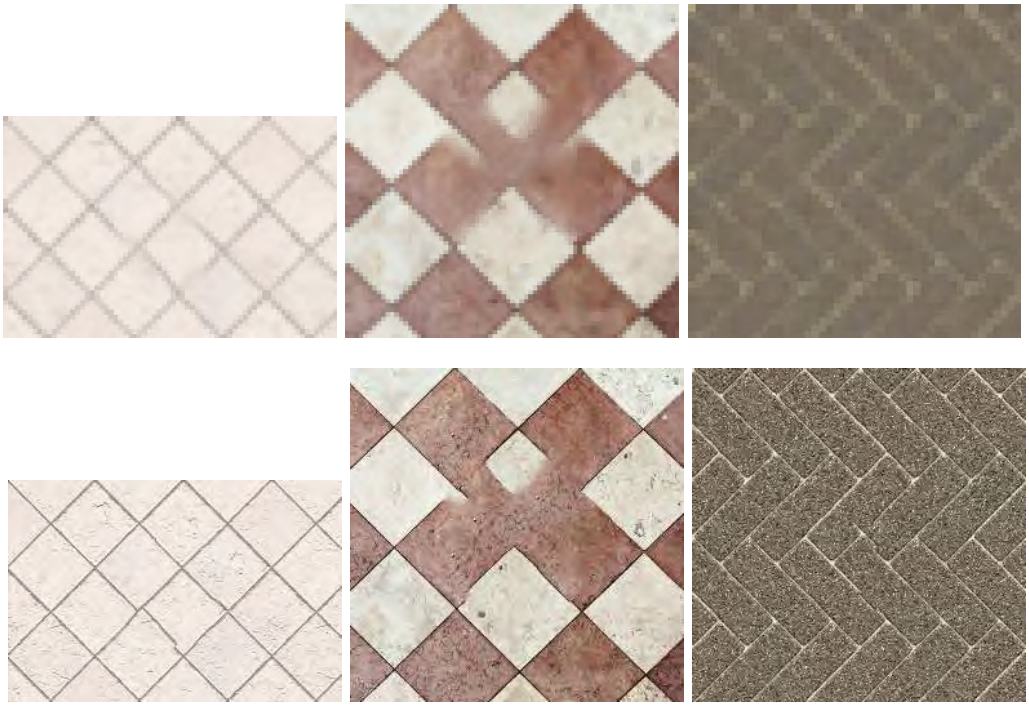


FIGURE 3.14 – Cas d'échec de NEWSON *et al.* [New+17]. Dans les trois cas, la structure n'est pas respectée. Deux causes peuvent expliquer cela. Pour les deux premières colonnes, l'initialisation s'est mal effectuée. Cela est dû à la mauvaise propagation des contours du carrelage durant la propagation par pelure d'oignon. Ensuite, l'initialisation se fait sur une résolution où certaines structures ne sont plus visibles (troisième colonne). Bien que l'initialisation soit correcte, un décalage au niveau de la structure apparaît au fur et à mesure que l'on remonte dans les résolutions supérieures. D'où la nécessité de choisir une résolution plus élevée pour initialiser.

dont la structure a été prise en compte dans la fonction de similarité. Cependant, bien que l'approche de Newson *et al.* soit globale dans la recherche de correspondance, elle reste locale dans l'initialisation. Pour cela, nous proposons une initialisation basée sur l'analyse des *offsets* qui assure, en plus de tenir compte de la structure et de la distribution du motif, une cohérence globale dans la zone complétée. Nous détaillons cette approche dans la section suivante.

3.3.2 Initialisation par *offsets*

Dans cette sous-section, nous détaillons notre initialisation via les *offsets*. L'algorithme 4 montre les modifications apportées dans l'*inpainting* basée sur *PatchMatch* en tenant compte des contraintes de structure. Les modifications sont écrites en rouge. Comme expliqué dans la section 2.4.2 du chapitre 2, un *offset* $\overrightarrow{o}$ est un vecteur de translation qui à un pixel p associe $p + \overrightarrow{o}$ tel que ce dernier soit le pixel qui, au delà d'un seuil τ minimise une fonction de similarité entre deux patchs centrés en p et q (*cf.* équation 3.7)

Par rapport à l'ensemble $\mathcal{O}$ des *offsets* d'une image I, nous considérons l'ensemble trié par ordre décroissant $\mathcal{O}_t$ selon la relation d'ordre :

$$\overrightarrow{o_1} \leqslant \overrightarrow{o_2} \Leftrightarrow occ(\overrightarrow{o_1}) \leqslant occ(\overrightarrow{o_2}), \tag{3.30}$$

où $occ(\overrightarrow{o})$ est l'occurrence de l'*offset* $occ(\overrightarrow{o})$.

$\mathcal{O}$ peut donc être considérée comme une liste triée des *offsets* $\mathcal{O}_t = [\overrightarrow{o_1}, \ldots, \overrightarrow{o_N}]$ avec $N = |\mathcal{O}_t|$. Nous avons des *offsets* qui représentent la structure globale mais aussi des *offsets* qui représentent le bruit de la texture. Les premiers ont une plus grande occurrence alors que les deuxièmes n'ont généralement une occurrence très faible. Pour réduire le temps d'exécution et améliorer la qualité de la coupure de graphe qui suit, on peut sélectionner un nombre d'*offsets* qui ont les occurrences les plus élevées. Cette dernière n'est effectuée ainsi qu'en tenant compte de ce sous-ensemble. Contrairement à [HS12b], nous ne prenons pas un nombre fixe d'*offsets* ($K = 50$). En effet, si nous avons une texture fortement régulière, nous n'aurons qu'une dizaine d'*offsets* de la première catégorie. Ajouter d'autres *offsets* « insignifiants » diminue la qualité de la coupure de graphe. Pour cela, nous ne prenons qu'une majorité $\mathcal{O}_m \subset \mathcal{O}_t$ d'*offsets*.

Nous considérons la population d'un sous-ensemble $\mathcal{O}' \subset \mathcal{O}_t$ définie comme telle :

$$\text{population}(\mathcal{O}') = \sum_{\overrightarrow{o} \in \mathcal{O}'} \text{occurence}(\overrightarrow{o}). \tag{3.31}$$

Nous définissons la λ-majorité $\mathcal{O}_m^\lambda \in \mathcal{P}(\mathcal{O}_t)$ comme :

$$\mathcal{O}_m^\lambda = [\overrightarrow{o_1}, \ldots, \overrightarrow{o_i}], \quad i \leqslant N, \quad \text{population}(\mathcal{O}_m^\lambda) \geqslant \lambda \ \text{population}(\mathcal{O}), \tag{3.32}$$

avec $\lambda \in [0, 1]$.

D'ici, nous effectuons une coupure de graphe [BK04 ; BVZ01] sur l'ensemble comprenant le masque et sa frontière extérieure $\overline{\Omega}$ en n'assignant comme label que les *offsets* de $\mathcal{O}_m^\lambda$. L'image de gauche de la figure 3.15 montre un exemple de carte de labels.

Influence de λ

Nous évaluons l'influence de λ, paramètre défini par l'équation 3.32, qui a un impact à la fois en terme de temps de calcul et de qualité du résultat. Le tableau 3.3 montre les temps de calcul de plusieurs textures de taille moyenne (600×800) pour différentes valeurs de λ. Nous avons naturellement un temps plus élevé pour des valeurs proches

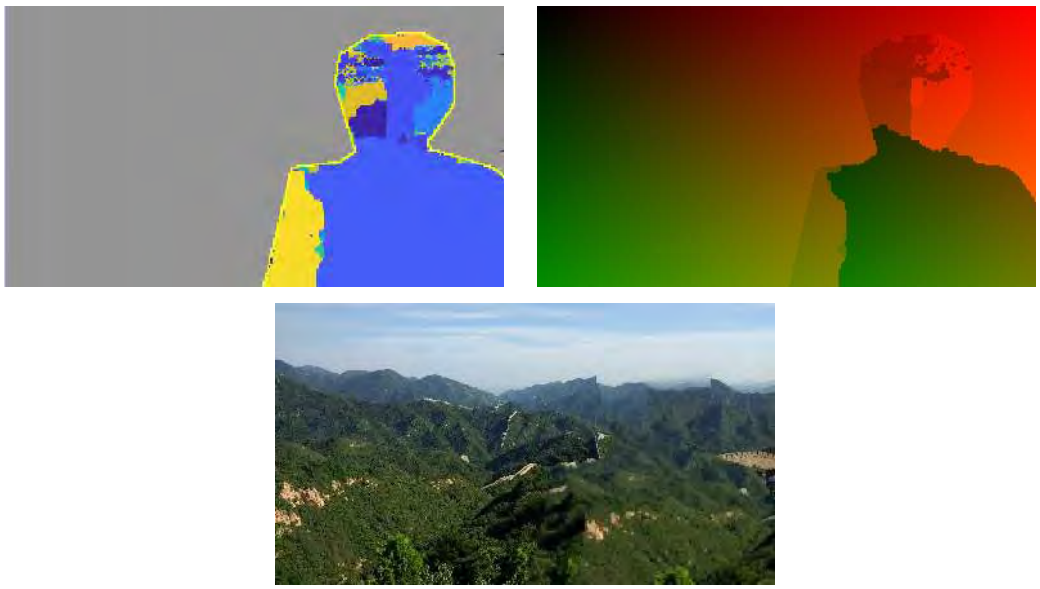


FIGURE 3.15 – Un exemple d'une carte de labels (première ligne, à gauche) et d'une carte de correspondances (première ligne, à droite) pour une image I avec son initialisation (deuxième ligne). Dans la première image, la couleur grise représente les pixels qui ne sont pas pris en compte pour la minimisation. Chaque pixel de Ω a une couleur correspond à un label, soit l'*offset* qui permet de compléter ce pixel. Dans cet exemple, une grande majorité de pixels sont assignés au même *offset* (couleur bleue). Dans la deuxième image, pour chaque pixel du masque, la couleur indique la position dans l'image du pixel correspondant selon l'*offset* attribué (dans cet exemple, les pixels images sont situés essentiellement à gauche et en haut du masque).

ratio λ	texture 28	texture 55	texture 43	texture 88
0.1	0.701 sec	0.834 sec	0.578 sec	1.167 sec
0.2	0.690 sec	0.799 sec	0.491 sec	1.171 sec
0.3	0.716 sec	0.841 sec	0.438 sec	1.313 sec
0.4	0.820 sec	0.794 sec	0.499 sec	1.497 sec
0.5	0.839 sec	0.727 sec	0.568 sec	1.405 sec
0.6	0.985 sec	0.780 sec	0.627 sec	1.708 sec
0.7	1.052 sec	0.760 sec	0.704 sec	1.735 sec
0.8	1.151 sec	0.911 sec	0.803 sec	1.795 sec
0.9	1.438 sec	1.060 sec	0.873 sec	2.241 sec
1.0	1.338 sec	1.277 sec	0.904 sec	2.312 sec

TABLE 3.3 – Temps de calcul en fonction du ratio de population sur plusieurs textures.

de 1.0. Cela est dû au fait que nous avons plus de labels à évaluer durant la phase de coupure de graphe.

La figure 3.16 nous donne des résultats de quelques textures du *benchmark* de données de la section 3.2.5 en fonction de λ. On peut y constater la difficulté de choisir de manière fiable λ : si λ est trop faible, on risque de ne pas avoir assez d'*offsets* ; si λ est trop élevé, des *offsets* non voulus (car ne représentant pas la structure du motif) peuvent être sélectionnés. Empiriquement, nous posons $\lambda = 0.75$ qui est bon compromis par rapport à la qualité des résultats.

Modification de la fonction de régularisation

Après avoir discuté de la population d'*offsets* à choisir, nous abordons la modification de la fonction de régularisation. Comme dans l'équation 2.31 vue dans le chapitre 2, nous devons minimiser l'énergie suivante pour chaque pixel $p \in \overline{\Omega}$ et chaque label l :

$$E = \sum_{p \in \overline{\Omega}} E_d(p, l) + \sum_{(p_1, p_2) \in \mathcal{N}} E_r(p_1, p_2, l_1, l_2), \tag{3.33}$$

où E_d est le terme d'attache aux données :

$$E_d(p, l) = \begin{cases} 0 & \text{si } I(p + \overrightarrow{o_l}) \notin \Omega \\ +\infty & \text{sinon} \end{cases} \tag{3.34}$$

et E_r est le terme de régularisation :

$$E_r(p_1, p_2, l_1, l_2) = |I(p_1 + \overrightarrow{o_{l_1}}) - I(p_1 + \overrightarrow{o_{l_2}})| + |I(p_2 + \overrightarrow{o_{l_1}}) - I(p_2 + \overrightarrow{o_{l_2}})|. \tag{3.35}$$

$\mathcal{N}$ est l'ensemble des paires de pixels voisins.

Nous changeons E_r, renommée E'_r afin de tenir compte de la structure de la texture dans la minimisation :

$$E'_r(p_1, p_2, l_1, l_2) = (1 - \alpha) E_r(p_1, p_2, l_1, l_2) + \alpha E_s(p_1, p_2, l_1, l_2), \tag{3.36}$$

avec le terme de régularisation E_r de l'équation 3.35 et

$$E_s(p_1, p_2, l_1, l_2) = ||\nabla I(p_1 + \overrightarrow{o_{l_1}}) - \nabla I(p_1 + \overrightarrow{o_{l_2}})|| + ||\nabla I(p_2 + \overrightarrow{o_{l_1}}) - \nabla I(p_2 + \overrightarrow{o_{l_2}})||. \tag{3.37}$$

E'_r mesure ainsi la régularisation avec un terme d'ordre 2.

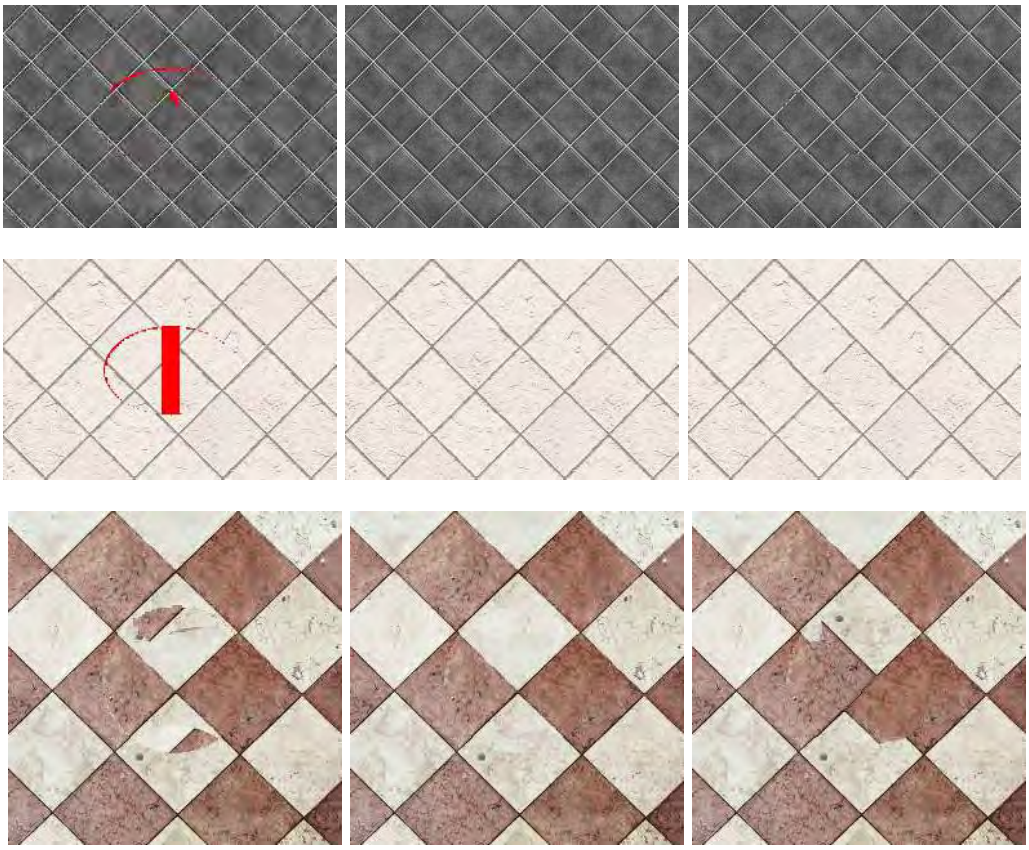


FIGURE 3.16 – Influence de ratio λ sur trois textures. Les λ pris pour la texture de la première ligne sont, de gauche à droite, 0.1, 0.6 et 1.0. Les λ pris pour la texture de la deuxième ligne sont, de gauche à droite, 0.5, 0.8 et 1.0. Les λ pris pour la texture de la troisième ligne sont, de gauche à droite, 0.1, 0.7 et 1.0. L'absence d'*offset* valable pour un pixel de Ω est symbolisé par la couleur rouge. Nous pouvons noter qu'une valeur faible engendre un faible nombre d'*offsets* ce qui peut se traduire par un manque de labels pour pouvoir compléter tous les pixels de Ω. À l'inverse, en considérant toute la population des *offsets* ($\lambda = 1.0$), nous avons des *offsets* indésirables et non représentatifs de la structure du motif qui sont utilisés pour minimiser l'énergie définie dans le chapitre 2. Cela fausse le résultat notamment dans le cas de la dernière ligne.

FIGURE 3.17 – Résultats finaux de deux textures complétées à partir de l'initialisation par *offsets*. La zone masquée et coloriée en rouge (première colonne) est complétée pour chaque texture avec différentes valeurs pour α (*cf.* équation 3.36). Dans la deuxième colonne, on prend $\alpha = 0$ pour les deux textures. Dans la troisième colonne, on prend $\alpha = 1$ pour les deux textures.

La pondération entre E_r et E_s via α permet de gérer des textures dont la tuile subit des variations colorimétriques dans l'image. Elle permet aussi d'accentuer les contours présents dans l'images. La figure 3.17 permet d'observer l'influence de α dans le calcul de l'initialisation et donc du résultat final. On constate bien que pour des textures ayant un motif répétitif soumis à des variations colorimétriques, le résultat est optimal pour un α proche de 1. Actuellement, le calcul de α n'est pas automatique. Cependant, le fait de choisir un α proche de 1 permet d'avoir des résultats corrects même pour des textures (généralement les textures ayant un motif non structuré) qui ont de bon résultats avec un $\alpha = 0$. En effet, avec un α proche de 1, on gomme les variations d'ordre 0 ou 1 de la texture. Et donc la régularisation sera moins sensible au variation de luminosité par exemple.

3.3.3 Analyse

L'initialisation par diffusion et par pelure d'oignons, bien qu'efficaces dans la majorité des cas, souffrent des limitations suivantes. Premièrement, la diffusion souffre d'un problème de "localité" et de propagation comme montré dans la figure 3.19. Le problème de localité est dû au fait que l'initialisation par diffusion est basée sur l'information située à la frontière extérieure $\partial\Omega_e$. Les pixels proches ont une valeur proche de ceux de la frontière tandis que les pixels éloignés de la frontière auront des valeurs qui s'éloignent de celles de la frontière. *PatchMatch* a donc plus de chance, dans le deuxième cas, de chercher une correspondance avec un pixel situé ailleurs que proche de $\partial\Omega_e$. Par conséquence, on aura deux régions voisines dans le masque où l'une, proche de la zone connue, aura une correspondance vers une zone proche tandis que l'autre aura une correspondance

Algorithm 4 Algorithme d'*inpainting* avec contraintes

Entrée: Image I, masque Ω, ratio d'initialisation r_{init}, étage de départ dans *PatchMatch* e_{PM}

Sortie: Image complétée I_c

Procédure InitialisationContraintes($I, \Omega, r_{init}, r_{PM}$)
  $I_{init}, labels \leftarrow$ Statistique+Graphcut(I, Ω, r_{init})
  Retourner $I_{init}, labels$
Fin Procédure
Procédure Inpaint(I, Ω, r_{PM})
  $I_{init}, c_{init} \leftarrow$ InitialisationContraintes($I, \Omega, r_{init}, r_{PM}$)
  Créer une pyramide multi-résolutions pour I and Ω avec N étages
  Pour $i \leftarrow e_{PM} : N$ **Faire**
    Si $i = e_{PM}$ **Alors**
      $c \leftarrow c_{init}$ ▷ Premier étage, on récupère la carte de correspondances créée par la fonction ci-dessus
    Sinon
      $c \leftarrow$ aggrandir($c, 2$) ▷ Récupérer la carte de l'étage précédent
    Fin Si
    AppliquerCorrespondances($I_{|\Omega}, c$)
    Pour $j \leftarrow 1 : n_{\text{PM}}$ **Faire**
      $c \leftarrow$ PatchMatch($I_{|\Omega}, I_{|O\setminus\Omega}, n_{\text{iter}}, c$)
    Fin Pour
  Fin Pour
Fin Procédure

pointant vers une zone située plus loin.

Le problème de propagation est dû au fait que les valeurs interpolées de même valeur vont généralement avoir une correspondance vers la même zone connue. On assiste à une duplication du même morceau connu de l'image dans la zone masquée. Cela donne un rendu final où un motif a été propagé de manière non contrôlé (voir les Figures 3.20 et 3.21).

L'initialisation par pelure d'oignon gère la deuxième limitation de la diffusion avec sa fonction de similarité qui tient compte de la structure du motif. Cependant, comme dans le cas de la diffusion, cette initialisation ne tient pas compte de la structure globale de la texture. Cela est dû à la complétion progressive. Celle-ci s'effectue à partir de la frontière du masque. La propagation de la structure se fait donc selon la normale du gradient du contour du masque. Il est probable qu'à la fin de l'initialisation, un *patch* copié au centre du masque ne respecte pas la structure de la texture.

Notre approche, à l'inverse, cherche des correspondances, dans toute la zone connue. De plus, les *offsets*, représentant la structure du motif, nous permettent d'avoir une initialisation basée sur des informations globales. *PatchMatch* peut ensuite régulariser les discontinuités existantes dans les résolutions supérieures (voir la deuxième colonne de la figure 3.20). Nous obtenons donc un rendu sans artefacts avec un respect global des structures présentes dans l'image (voir deuxième colonne de la figure 3.21).

Remarque sur le choix de la résolution pour chercher les *offsets* : comme nous montre la troisième image de la figure 3.14, une bonne initialisation n'entraîne pas forcément une bonne complétion. En effet, on constate que l'initialisation calculée respecte la structure de la texture à la résolution considérée. Cependant, en raffinant via *PatchMatch*, on note un décalage au centre de l'image. Cela est dû au fait que la structure, bien que

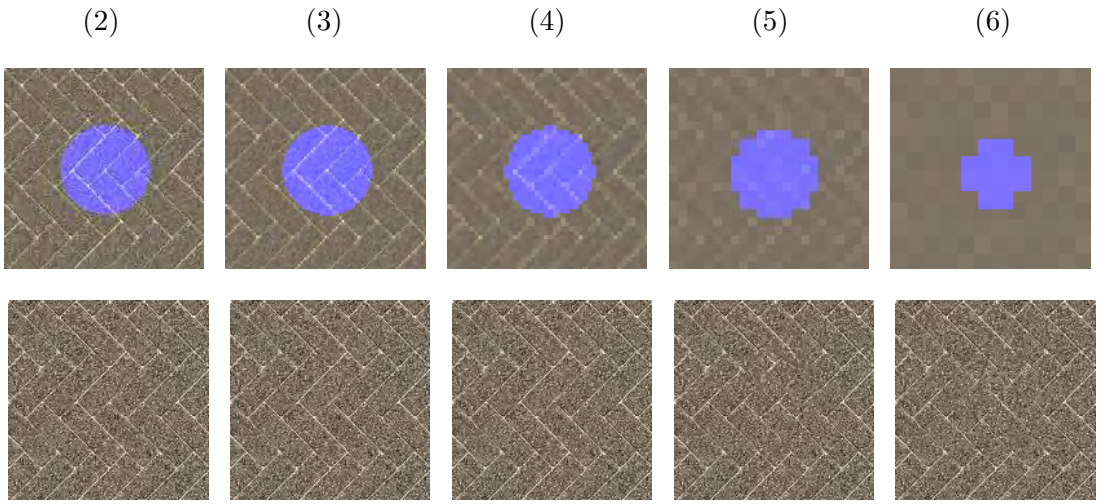


FIGURE 3.18 – Impacte de la résolution sur l'analyse des *offsets* et du respect de la structure dans les résolutions supérieures. Première ligne : initialisation à la résolution 2^n fois plus petite par rapport à la résolution initiale (n indiqué au-dessus de l'image). La zone masquée, qui a été initialisée, est représentée en bleu. Deuxième ligne : rendu final, obtenu à partir de l'initialisation de le première ligne. On constate la présence de décalages dans les rendus finaux quand l'analyse des *offsets* est effectuée dans des très basses résolution. À l'inverse, on ne voit pas de décalages quand l'analyse est faite à des résolutions intermédiaires.

correctement devinée, reste grossière. Si la structure n'est pas recontrôlée dans les étages plus haute résolution de la pyramide multi-résolutions, cela peut engendrer des décalages, comme illustré.

Notre approche subit également cet effet de décalage selon le niveau où l'on recherche les *offsets*. La figure 3.18 nous montre un exemple avec des initialisations à différents niveaux de résolution. On peut voir que plus la résolution d'analyse est faible, plus l'estimation des structures dans les résolutions supérieures est difficile. En terme de compromis entre la recherche rapide des fortes structures et la gestion des décalage, nous proposons empiriquement une résolution 2 fois plus petite à celle de l'image d'origine. Le choix de la résolution est débattu dans les perspectives.

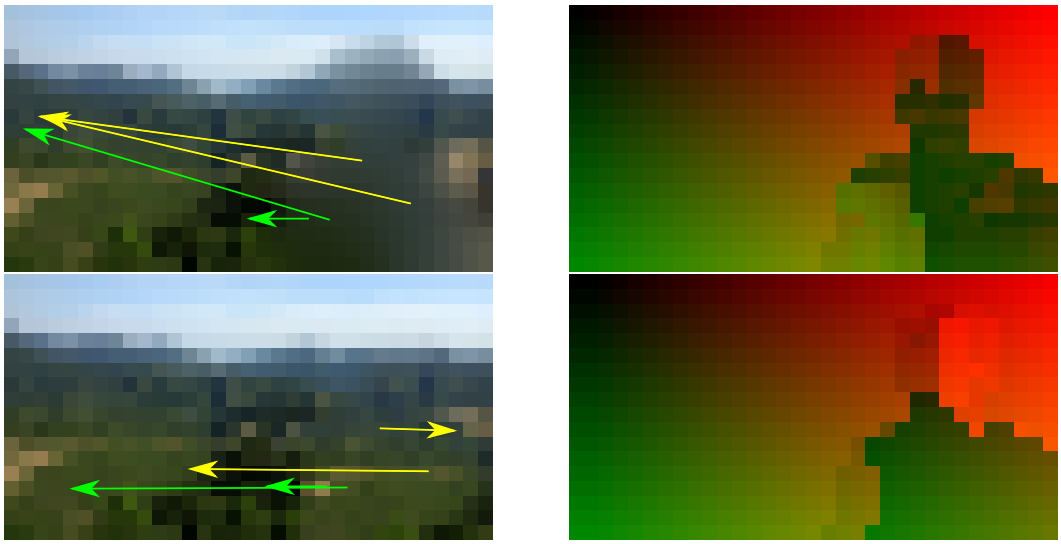


FIGURE 3.19 – Cartes de correspondances calculées à partir d'une initialisation par diffusion (première ligne) et par *offsets* (deuxième ligne). La carte de correspondances (droite) est calculée via *PatchMatch* à partir de l'initialisation (gauche). Les flèches de couleur indiquent les données qui seront copiées dans le masque Ω à l'étage de résolution suivante. Le couleur verte indique le problème de localité concernant la correspondance tandis que la couleur jaune indique le problème de propagation des données connues. *PatchMatch*, approche de recherche de correspondance globale, va chercher des correspondances entre un masque initialisée à partir d'informations locales et le reste de l'image. Dans le cas de l'initialisation par diffusion, les valeurs des pixels sont initialisées par interpolation et non par copie. *PatchMatch* va donc trouver une association avec un pixel dont le *patch* associé sera le plus similaire. D'un côté les pixels proches de la frontière du masque vont avoir des valeurs proches des pixels de la frontière de masque ; *PatchMatch* va leurs trouver une correspondance avec des pixels proches. De l'autre, les pixels au milieu du masque, ont une valeur éloignée de ceux qui sont proches de la frontière ; *PatchMatch* va leurs trouver une correspondance dans une zone similaire mais qui se situe loin du masque (flèches vertes).
De même, les pixels situés sur la zone au centre du masque ont des valeurs proches les uns des autres. Il est donc probable que *PatchMatch* crée une correspondance pointant vers la même zone connue. Cela engendre une propagation non contrôlé d'un même motif (flèches jaunes).
À l'inverse, dans notre approche, l'initialisation générée par les *offsets* va forcer *PatchMatch* à chercher dans la zone où les valeurs des pixels ont été copiées durant l'initialisation. L'impact se fait sentir dans la carte de correspondance. En effet, le vecteur liant un pixel du masque à son pixel correspondant suit la direction d'un des *offsets* calculés (ici les *offsets* dominants sont les directions horizontales).

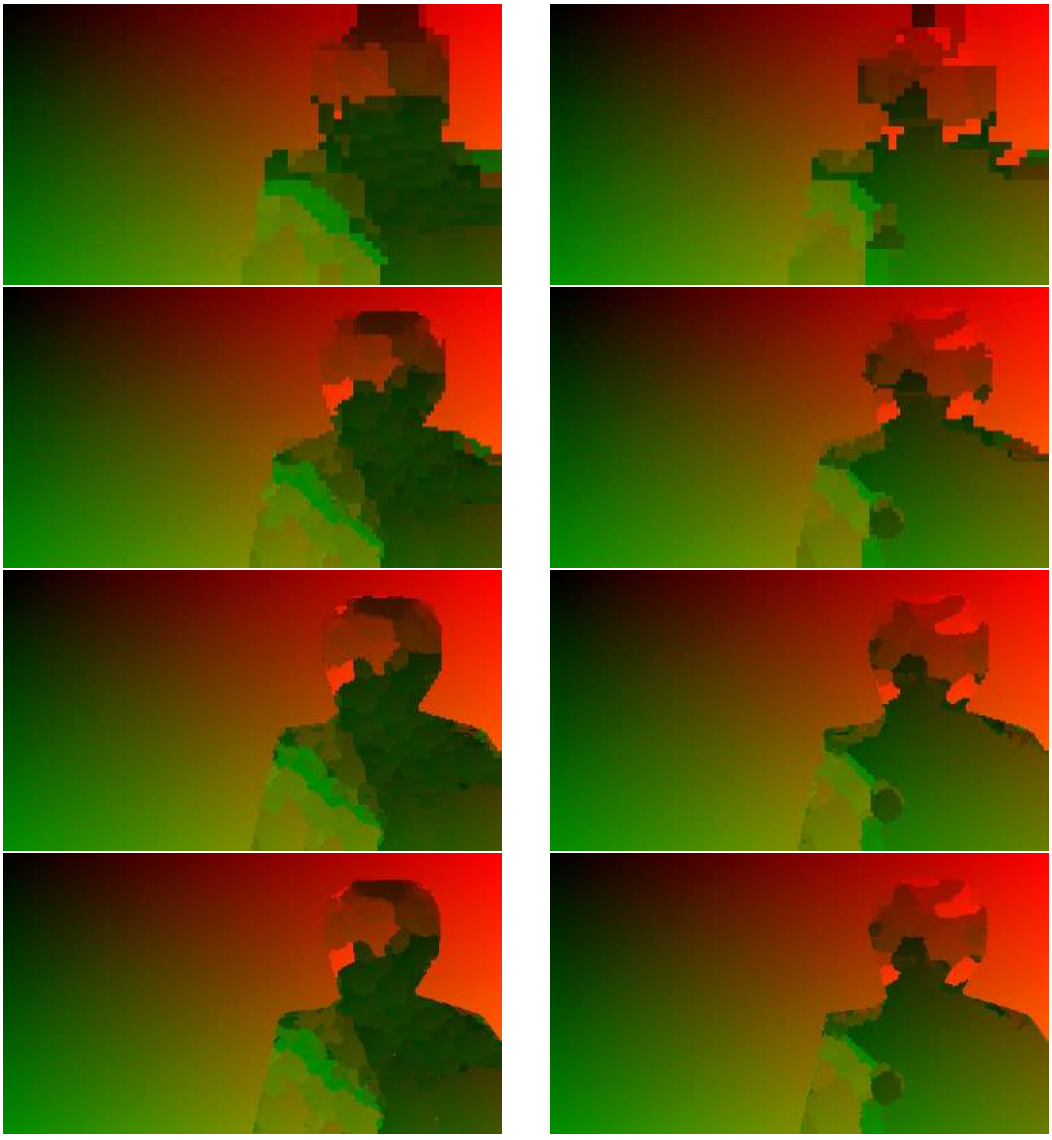


FIGURE 3.20 – Carte de correspondances c^k à différentes étages k, de l'étage de plus faible résolution (ligne du haut) à l'étage de plus forte résolution (ligne du bas) de la pyramide de résolution avec l'initialisation par diffusion (colonne de gauche) et l'initialisation par *offsets* (colonne de droite). Les images associées sont affichées dans la figure 3.21. On constate que dans le cas de *PatchMatch* avec l'initialisation basée diffusion, plusieurs zones du masque ont leurs pixels qui ont une correspondance pointant vers la même zone (couleur verte foncée). Dans notre approche d'initialisation par les *offsets*, la carte de correspondance possèdent moins de discontinuité. Il y a donc une propagation contrôlée selon la direction d'un *offset* (ici la direction horizontale).

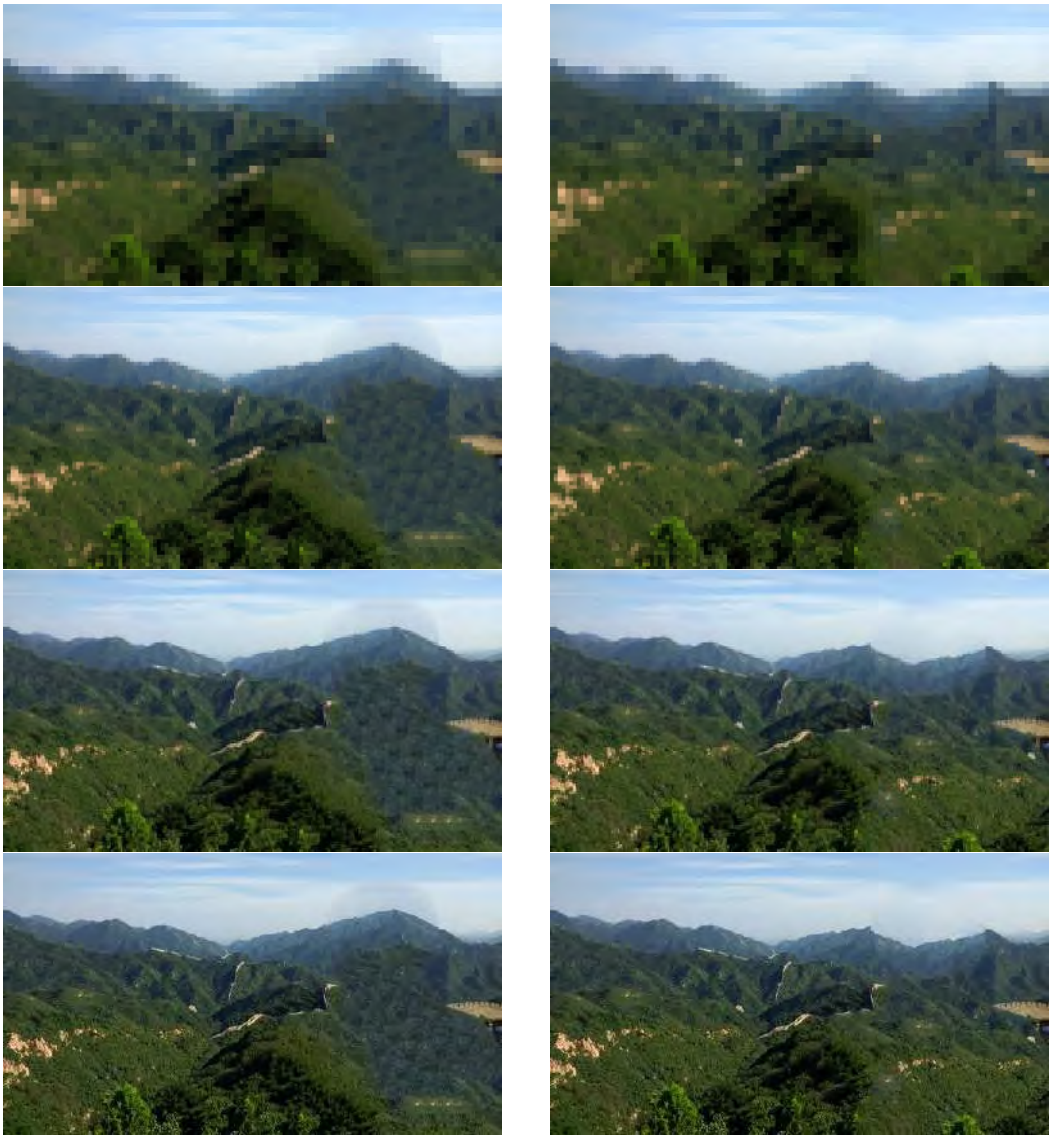


FIGURE 3.21 – Image I^k à différentes étages k de la pyramide de résolution avec l'initialisation par diffusion (colonne de gauche) et l'initialisation par *offsets* (colonne de droite) à partir des cartes de correspondances de la figure 3.20.

image	*PatchMatch*	Newson *et al.* [New+17]	méthode proposée
carrelage	6.691 sec	37.294 sec	7.080 sec
briques	7.279 sec	74.455 sec	6.001 sec
arbre	6.983 sec	31.495 sec	4.245 sec
saut	4.572 sec	37.778 sec	12.535 sec
paysage	9.654 sec	75.103 sec	6.810 sec
granite	13.144 sec	100.824 sec	7.038 sec
phare	5.298 sec	25.064 sec	3.218 sec

Figure 3.22 – Tableau du temps de calcul des trois approches comparées. Deuxième colonne : approche d'*inpainting* basée sur *PatchMatch*. Troisième colonne : approche avec initialisation par pelure d'oignon de Newson *et al.* [New+17]. Quatrième colonne : notre approche avec initialisation par *offsets* avec la régularisation d'ordre 2.

3.4 Expérimentations

Dans cette section, nous présentons quelques résultats de notre approche basée sur une initialisation par *offsets* et sur une fonction de régularisation d'ordre 2 dans la coupure de graphe appliquée à différents types de textures.

Nous testons cette approche de contraintes par *offsets* sur le *benchmark* personnalisé de la section 3.2.5 (environ 200 images). De plus, des photos sont ainsi testées pour montrer la robustesse dans des cas non contrôlés, c'est-à-dire des images où non seulement la structure du motif varie, mais aussi sa distribution ainsi que sa variété (plus ou moins loin, orienté différemment). Dans la deuxième sous-section nous comparons nos résultats avec l'approche *PatchMatch* classique et l'approche de Newson *et al.* [New+17].

3.4.1 Implémentation et temps de calcul

La méthode proposée est codée en `C++`. Nous utilisons la bibliothèque CImg [Tsc12] pour la gestion et le traitement des images. Nous utilisons l'implémentation de l'algorithme de correspondance *PatchMatch* de cette bibliothèque. Nous utilisons la bibliothèque GCO pour la minimisation d'énergie par coupure de graphe pour l'initialisation par *offsets*. Nous avons recodé l'approche d'*inpainting* basée sur *PatchMatch* (à ne pas confondre avec l'algorithme *PatchMatch* déjà fourni par CImg) puis l'avons modifié pour tenir compte de l'initialisation calculée en première partie.

Le tableau 3.22 donne un temps de calcul des trois approches comparées sur plusieurs exemples montrés dans ce chapitre. Nous avons choisi des images de texture par notre ensemble de textures qui posent problème à l'approche d'*inpainting* basé sur *PatchMatch* ainsi qu'à l'approche de Newson *et al.* (initialisation contrôlée par pelure d'oignon). L'exécution a été effectuée sur une machine Linux Mint machine équipée d'un processeur i7-6700HQ cadencé à 2.6 GHz et avec 8 GB de RAM. Il n'y a pas d'optimisation GPU. Cependant, il est possible d'implémenter sur un GPU la section concernant la recherche de correspondance entre les pixels de la zone masquée et ceux de la zone connue en utilisant l'algorithme *jump flood* [RT06]. La recherche des *offsets* ainsi que la mise à jour via *PatchMatch* est multi CPUs en utilisant OpenMP.

Nous constatons que notre approche a généralement un temps de calcul équivalent à celui de *PatchMatch* (moins de 7.080 sec par image). Notre approche est aussi beaucoup plus rapide que la méthode de Newson *et al.* [New+17]. Notre approche, par sa vitesse d'exécution, est intéressant dans le cadre de ressources limitées sur système mobile.

image	*PatchMatch*	Newson	méthode proposée
texture 60	0.9404	0.9498	0.9554
texture 27	0.7899	0.8016	0.8027
texture 84	0.8392	0.8167	0.8121
texture 0	0.9116	0.9174	0.9142
texture 2	0.8817	0.8899	0.8821
texture 5	0.8702	0.8527	0.8642

FIGURE 3.23 – Tableau des mesures selon la SSIM des trois approches comparées. Deuxième colonne : approche d'*inpainting* basée sur *PatchMatch*. Troisième colonne : approche avec initialisation par pelure d'oignon de NEWSON *et al.* [New+17]. Quatrième colonne : notre approche avec initialisation par *offsets* avec la régularisation d'ordre 2. Chaque texture est complétée selon un masque circulaire créé au centre de l'image. Plus une valeur est proche de 1, plus la texture complétée est proche de la vérité-terrain. Les textures sont numérotés selon leur indexation dans le *benchmark* personnalisé.

3.4.2 Rendu final

Les figures 3.24 à 3.27 sont organisés en groupes de quatre images de la manière suivante :

— haut-gauche : Vérité terrain avec zone masquée en rouge
— haut-droite : Résultats de *PatchMatch* [Bar+09]
— bas-gauche : Résultats de NEWSON *et al.* [New+17]
— bas-droite : Notre approche d'initialisation par *offsets* avec la régularisation d'ordre 2

L'évaluation des résultats d'*inpainting* reste délicate dans la mesure où il n'existe pas de mesure quantitative fiable. Le PSNR, par exemple, se base sur la moyenne aux moindre carrées ; elle ne représente généralement pas la perception de la qualité par l'œil humain [HG08]. La SSIM, mesure la similarité au niveau de la structure. Pour cela, elle compare l'intensité des pixels selon la luminance et le contraste. Dans notre contexte de respect de la structure dans l'*inpainting*, nous effectuons une SSIM sur quelques textures de notre *benchmark*. Le tableau 3.23 montre les mesures effectuées sur les résultats des trois méthodes utilisées par ces deux métriques. On peut noter que les approches *PatchMatch* et de NEWSON *et al.* [New+17] ont une mesure meilleure que la nôtre pour la texture 2 (texture du bas de la figure 3.24) et la texture 0 (texture du bas de la figure 3.26). Pourtant, la qualité du rendu effectuée est clairement meilleure que celle des deux autres approches. Nous nous baserons dans ce document sur une évaluation purement qualitative.

Remarque 3.4.1. ISOGAWA *et al.* [Iso+18] proposent une nouvelle méthode d'évaluation quantitative qui crée un modèle d'entraînement pour estimer la qualité du résultat de l'*inpainting*. Celui-ci se sert de la similarité comme relation d'ordre. Des niveaux de dégradation sont créés en prenant par exemple non pas le meilleur *patch* mais le n-ième *patch* similaire. Evaluer notre méthode avec cette mesure est une perspective de cette thèse.

Sur les Figures 3.24 à 3.26 qui contiennent des images de texture avec des motifs respectivement structurés et irréguliers et semi réguliers, nous constatons que notre approche donne les meilleurs résultats. Dans tous les cas, l'approche classique *PatchMatch* ne s'en sort que pour les textures irrégulières. L'approche de NEWSON *et al.* [New+17] donne des résultats satisfaisants sur la texture de brique de la figure 3.24 et sur les tex-

tures irrégulières de la figure 3.25, et notre approche donne des résultats relativement similaires dans ces cas mais sensiblement meilleurs dans le cas de la figure 3.25. Dans le cas des textures difficiles type carrelage montrées dans la figure 3.26, on constate que notre méthode sort du lot par rapport à l'approche de NEWSON *et al.*.

Bien que cela soit à la limite du contexte de la thèse, nous avons quand même testé notre approche sur des images de paysage comme montré dans la figure 3.27. Nous constatons que notre approche complète ces images de manière respectueuse de leurs contenu et tout à fait satisfaisante alors que l'approche de NEWSON *et al.* échoue à compléter convenablement.

Notons enfin les résultats de la figure 3.27 où un masque de grande taille est dessiné. Nous constatons que notre approche remplit de manière correcte l'intérieur du masque, bien que la qualité du résultat soit inférieure à celle d'un résultat où un masque de petite taille est dessiné. À l'inverse, *PatchMatch* et surtout NEWSON *et al.* échouent à compléter fidèlement le masque.

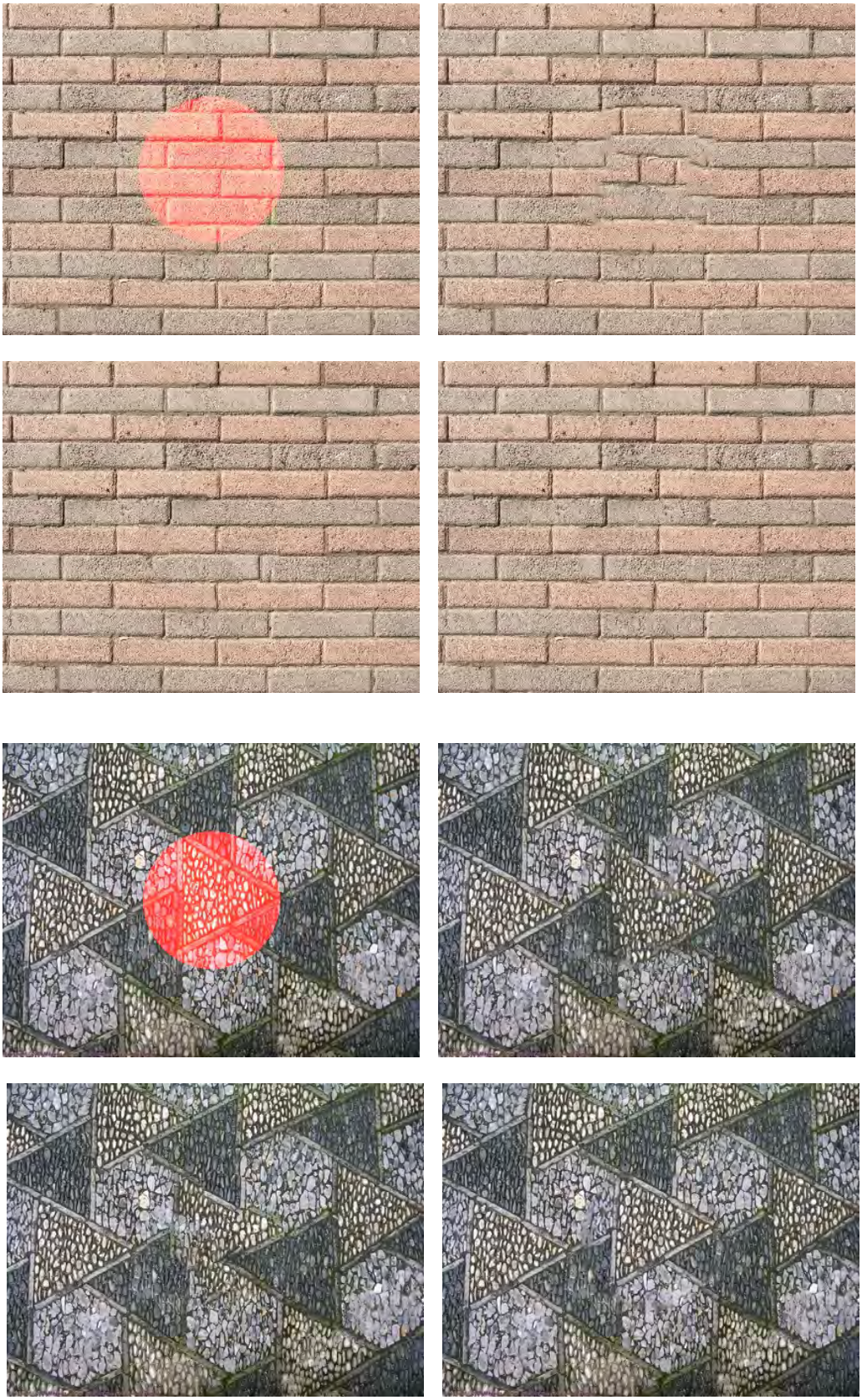


FIGURE 3.24 – Résultats sur des textures régulières d'intérieur. Chaque bloc de quatre images représentant une texture. En haut à gauche, la vérité terrain avec la zone masquée (coloriée en rouge). En haut à droite, *inpainting PatchMatch* [Bar+09] classique (initialisation basée diffusion). En bas à gauche, l'approche de NEWSON *et al.* [New+17]. En bas à droite, notre approche. Pour la première texture, nous constatons que *PatchMatch* échoue à compléter correctement alors que NEWSON *et al.* et notre approche complètent convenablement. Pour la deuxième texture, nous constatons que *PatchMatch* et NEWSON *et al.* échouent à compléter correctement alors que notre approche complète convenablement.

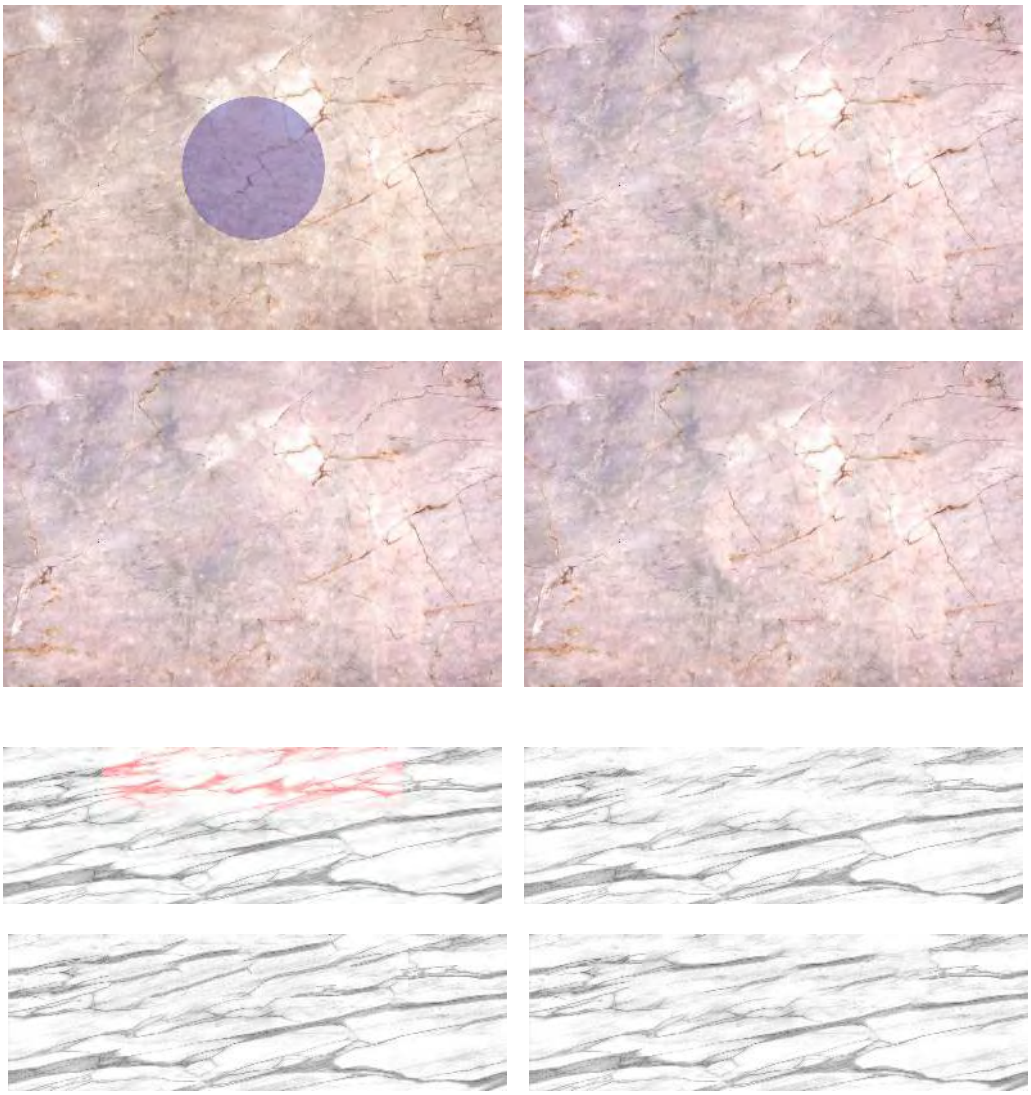


FIGURE 3.25 – Résultats sur des textures irrégulières d'intérieur. En haut à gauche, la vérité terrain avec la zone masquée (coloriée en bleu pour la première texture, coloriée en rouge pour la deuxième texture). En haut à droite, *inpainting PatchMatch* classique (initialisation basée diffusion). En bas à gauche, l'approche de NEWSON *et al.* [New+17]. En bas à droite, notre approche. Pour les deux textures, les trois approches fournissent un résultat convenable. Notons cependant que l'approche de NEWSON *et al.* et la nôtre fournissent un résultat plus réaliste dans le cas de la deuxième texture (la structure noire horizontale est mieux propagée).

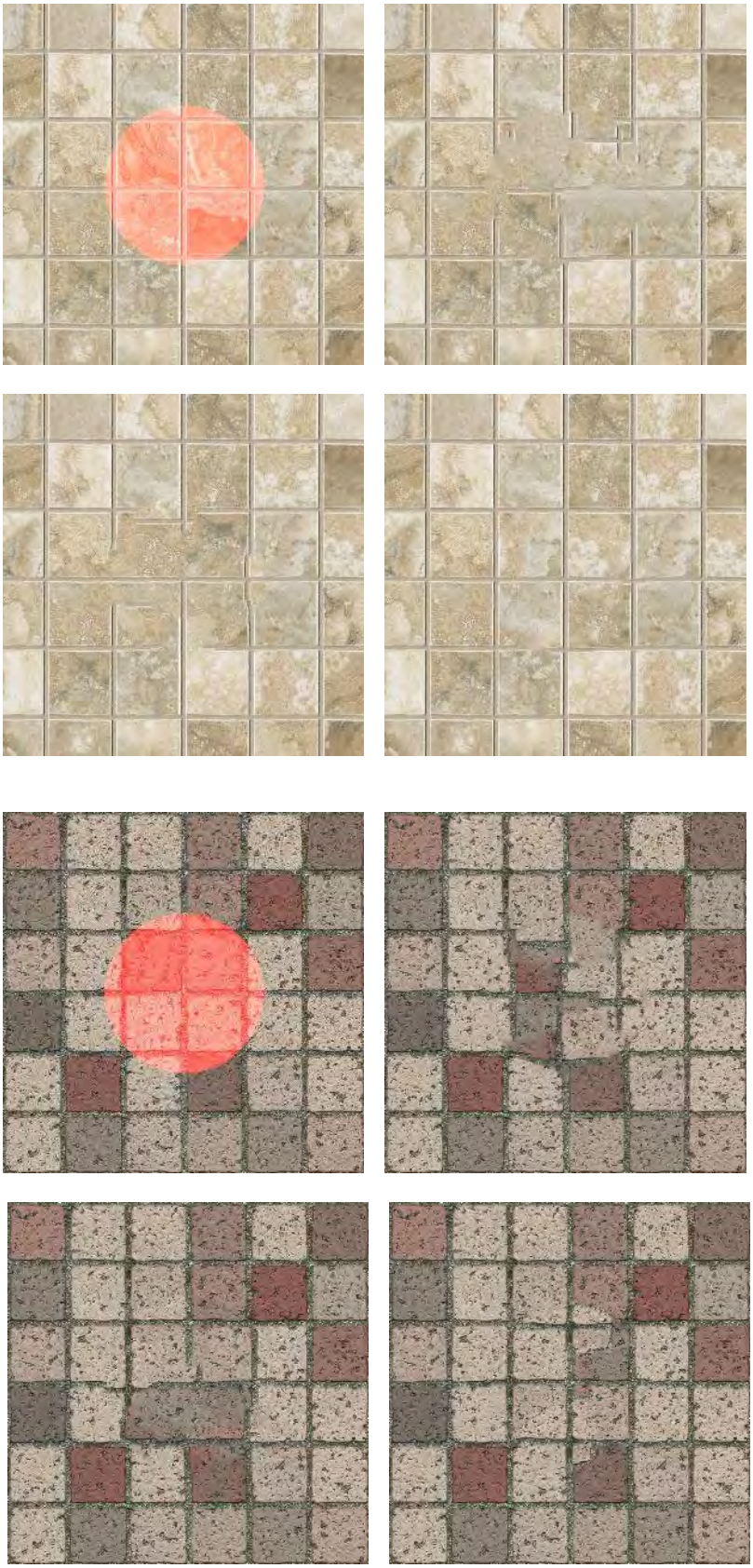


FIGURE 3.26 – Résultats sur des textures difficiles. En haut à gauche, la vérité terrain avec la zone masquée (coloriée en rouge). En haut à droite, *inpainting PatchMatch* classique (initialisation basée diffusion). En bas à gauche, l'approche de NEWSON *et al.* [New+17]. En bas à droite, notre approche. Pour la première texture, nous constatons que *PatchMatch* et NEWSON *et al.* échouent à compléter correctement alors que notre approche complète convenablement. Concernant la deuxième texture et dans le cas de notre approche, bien que la couleur ne soit pas totalement respectée au sein d'un carreau, la structure du motif est respectée.

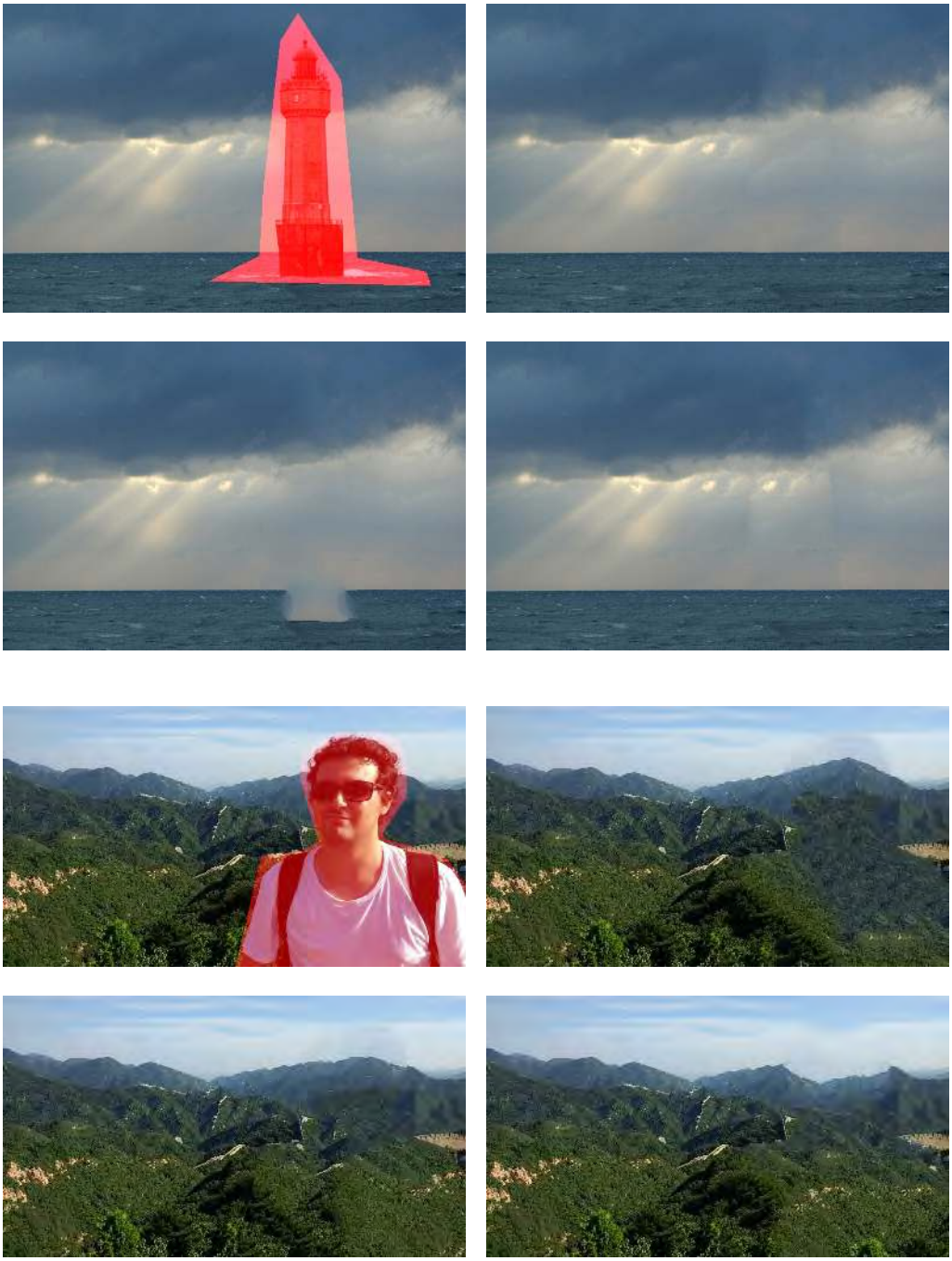


FIGURE 3.27 – Résultats sur des photos variées (ici paysage d'extérieur). Chaque bloc de quatre images représentant une texture. En haut à gauche, la vérité terrain avec la zone masquée (coloriée en rouge). En haut à droite, *inpainting PatchMatch* classique (initialisation basée diffusion). En bas à gauche, l'approche de NEWSON *et al.* [New+17]. En bas à droite, notre approche. Nous constatons que notre approche arrive de manière honorable à effacer des éléments d'une image. À l'inverse, NEWSON *et al.* n'a pas réussi à initialiser correctement l'horizon d'où un résultat médiocre.

FIGURE 3.28 – Résultats sur des photos variées (paysages). En haut à gauche, la vérité terrain avec la zone masquée (coloriée en rouge). En haut à droite, *inpainting PatchMatch* classique (initialisation basée diffusion). En bas à gauche, l'approche de NEWSON *et al.* [New+17]. En bas à droite, notre approche. Sur des photos variées, nous pouvons constater que notre approche fournit un résultat équivalent à celui de NEWSON *et al.*.

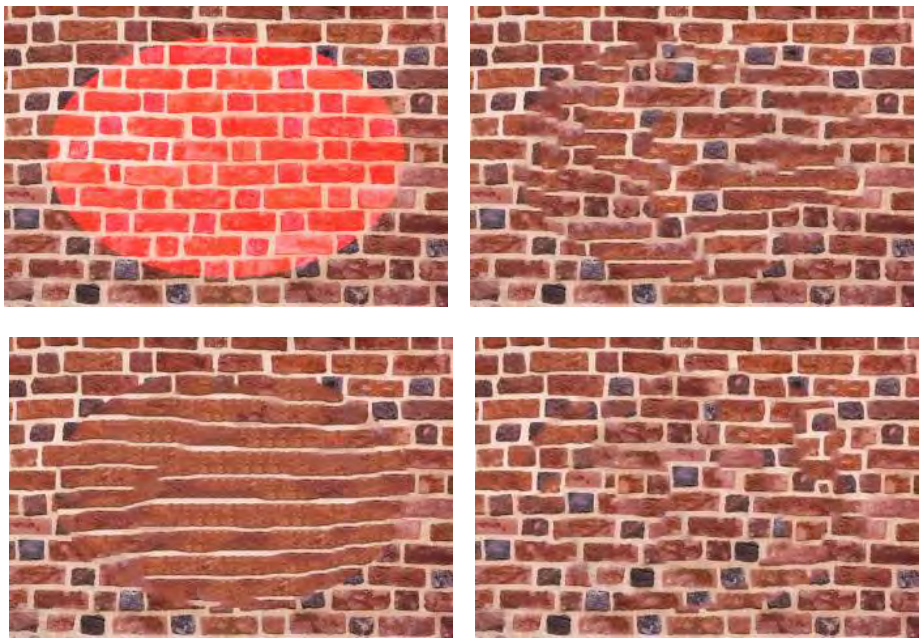


FIGURE 3.29 – Résultat sur une texture avec un masque de grande taille. En haut à gauche, la vérité terrain avec la zone masquée (coloriée en rouge). En haut à droite, *inpainting PatchMatch* classique (initialisation basée diffusion). En bas à gauche, l'approche de NEWSON *et al.* [New+17]. En bas à droite, notre approche. Remarquons que dans le cas d'une grande zone à compléter, *PatchMatch* et surtout NEWSON *et al.* n'arrivent pas compléter avec le peu d'information disponible. À l'inverse notre approche arrive à produire un résultat meilleure mais qui reste imparfait par rapport à une situation où la zone à compléter est plus petite.

3.5 Conclusion et perspectives

Dans cette section nous présentons les limites de notre approche ainsi que des éléments de résolutions possibles.

3.5.1 Gestion des *offsets* indésirables

Problématique

Notre approche dépend fortement des *offsets* calculés durant la première étape du calcul. Il peut arriver d'obtenir des *offsets* qui ne reflètent la répétition de la structure de la texture et pourtant d'occurrence suffisamment importante pour être pris en considération. Cela se produit généralement quand la structure ne ressort pas suffisamment par rapport à la couleur, les *offsets* sont sélectionnés pour des similarités de couleur et non de texture. Aussi, si nous devons compléter une image avec peu de texture, l'analyse statistique sera de mauvaise qualité et des *offsets* non désirables risque d'être sélectionnés.

Si ces *offsets* sont gardés en tant que label pour la minimisation de l'énergie décrite dans l'équation 3.33, la carte de correspondances transmise à *PatchMatch* ne sera pas optimale.

offset	occurrence
$(0, 22)$	22
$(0, 24)$	18
$(0, 124)$	14
$(-54, 72)$	6

FIGURE 3.30 – Tableaux d'*offsets* de l'image de figure 3.31 classés de manière décroissante selon leur occurence. Seul les *offsets* dominants ont été écrits. Les *offsets* incorrectes sont écrits en rouge.

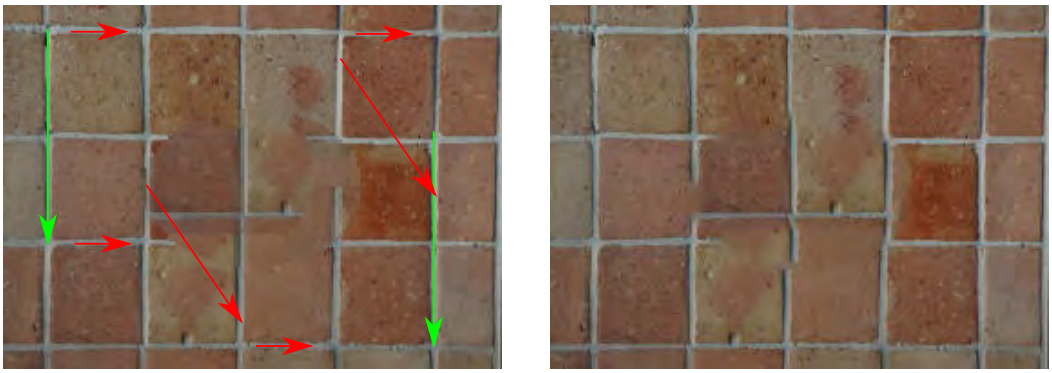


FIGURE 3.31 – Cas d'échec d'une texture. Gauche : initialisation avec les *offsets* calculés représentés par des flèches. Droite : rendu final. Les *offsets* trouvés sont dans le tableau 3.30. Les flèches rouges représentent les *offsets* non voulus. Les flèches rouges représentent les *offsets* qui respectent la structure de la texture.

Les Figures 3.31 et 3.30 montrent le tableau des *offsets* de plus forte occurrence ainsi que l'initialisation d'une image. Nous constatons que l'initialisation nous donne un résultat qui n'est pas conforme au motif du carrelage.

Éléments de résolution

Généralement, une texture est composée de plusieurs gammes de fréquences :
— les basses fréquences représentent la couleur ainsi que la structure globale de la texture
— les hautes fréquences représentent le motif ainsi que le détail de la texture

Actuellement, le fait d'analyser les *offsets* en basse résolution est équivalent à effectuer un passe-bas sur l'image. Or, on pourrait, à la place, effectuer plusieurs passes-bande selon une bande-passante à définir. On effectuerait notre analyse d'*offsets* sur chaque image filtrée. Une idée serait alors de diviser I en un ensemble d'images grâce à une transformation par vaguelettes *a trous* [FU99] et de traiter l'initialisation sur l'ensemble d'images et de ne considérer que les *offsets* qui ne ressortent que dans la majorité des images. Enfin, une autre idée serait d'améliorer l'énergie de minimisation, notamment au niveau du terme de régularisation. Pour cela, le terme d'ordre 2 ne serait plus basée sur le calcul de gradient. Il serait basé plutôt sur le calcul de tenseurs de structure qui fournit une information plus complète à propos de la structure de la texture.

3.5.2 Conclusion

Nous avons abordé dans ce chapitre le cas de la structure de la texture et comment gérer les différents cas possibles pour effectuer une complétion optimale. Nous avons d'abord tenté de distinguer les textures ayant une structure suffisamment régulière qui nécessite une approche d'*inpainting* différente des autres textures, qui peuvent être facilement complétée par une approche d'*inpainting* par *offsets*.

Cette classification permet de traiter de façon satisfaisante les textures qui sont clairement d'un type identifié, soit par un *inpainting* de type, *PatchMatch* soit par un *inpainting* de type *offsets*, Par contre certaines textures semi-régulières ne sont bien traités par aucune des deux approches d'*inpainting* considérées. Nous avons donc développé une approche basée sur une meilleure initialisation qui combine les deux méthodes détaillées dans le chapitre 2 qui améliore les résultats par rapport à ces deux méthodes et leurs améliorations proposées dans l'état de l'art.

Chapitre 4

Critère de confiance radiométrique pour l'*inpainting* basé *patchs*

Sommaire

4.1 Introduction

Dans ce chapitre, nous abordons le passage à la 3D de l'*inpainting* qui a été considéré sur des images 2D dans les chapitres précédents. Comme nous avons vu dans le chapitre 1, le fait d'appliquer un processus d'*inpainting* directement à l'image de la prise de vue ne fournit que des résultats médiocres. Cela pose des problèmes de segmentation notamment à la frontière entres les plans de la scène. Aussi, il y a le problème d'échelle des motifs de la texture : le même motif sera déformé dans une image selon sa position dans la scène 3D (plus petit car loin, les lignes parallèles du motif ne sont parallèles en raison de la déformation de perspective). Pour contrer cela et améliorer le rendu fourni par

l'*inpainting*, des approches comme Kawai *et al.* [KSY15] et Siltanen [Sil15] changent l'espace de traitement en exploitant les caractéristiques d'intérieur : au lieu d'appliquer directement l'*inpainting* sur l'image rendue de la scène, ils utilisent l'avantage d'avoir une géométrie planaire par morceaux pour mieux guider le processus d'*inpainting*. En effet, ils calculent, pour chaque prise de vue, une matrice de transformation appelée homographie entre le plan de la caméra et le plan 3D considéré (sol, mur). Cette transformation, appelée rectification, est appliquée à la région d'un plan visible dans l'image. Ainsi, la texture retrouve sa régularité en terme de motif et d'échelle. Cependant, les méthodes d'*inpainting* classiques ne tiennent pas compte la variation de la qualité des données de l'image dans l'espace rectifié. Cela peut générer des résultats d'*inpainting* dégradés. Nous proposons de quantifier la qualité des données rectifiées. Pour cela un critère de confiance est estimé sur l'image rectifiée et nous l'utilisons pour améliorer le processus d'*inpainting*.

Le chapitre est organisé de la façon suivante. Tout d'abord, nous rappelons quelques éléments de vision par ordinateur concernant le modèle classique de caméra ainsi que la définition de l'homographie. Ensuite, analysons la qualité de l'information durant la rectification ainsi que les conséquences dans le processus d'*inpainting* si la variation de résolution n'est pas prise en compte. Nous introduisons ensuite notre critère de confiance qui est basé sur des critères radiométriques (dont les notions sont rappelées en début de section). Nous présentons des applications de ce critère notamment en terme d'*inpainting* où les approches détaillées dans le chapitre 2 comme l'approche basée sur *PatchMatch* ou l'approche par *offsets* sont modifiées pour tenir compte de ce critère ; un flou avec un noyau variable est également introduit. Nous montrons l'efficacité de la prise en compte de ce critère dans une approche d'*inpainting* basée *patchs* à travers des résultats sur des scènes de synthèse ainsi que dans des situations réelles où ces modifications sont appliquées. Nous concluons en énonçant les perspectives possibles.

4.2 Homographies : rappels et utilisation

Dans cette section, nous rappelons le modèle de la caméra, la notion d'homographie et son application dans le cadre de l'élimination de perspective.

4.2.1 Les coordonnées homogènes

Pour rappel, les coordonnées homogènes sont un outil de base en vision par ordinateur, notamment pour caractériser les espaces projectifs. Elles reposent sur une notation dans laquelle un vecteur d'un espace de dimension n est représenté par un vecteur de dimension $n+1$. Dans le cas de l'espace vectoriel $\mathbb{R}^n$, on appelle $\mathbb{P}^n = \mathbb{R}^{n+1} \setminus \{0\}$ son espace projectif associé. Deux points $\mathbf{M}, \mathbf{N} \in \mathbb{P}^n$ dénotent le même point de l'espace projectif si leurs coordonnées sont proportionnelles à un scalaire λ non nul près : $\mathbf{M} = (M_1, M_2, \ldots, M_n) = \mathbf{N} = (\lambda M_1, \lambda M_2, \ldots, \lambda M_n)$, avec $\lambda \neq 0$. Un point de l'espace projectif $\mathbb{P}^n$ est dit "normalisé" si sa dernière cordonnée est 1. Dans ce cas, les autres coordonnées correspondent alors aux coordonnées du point l'espace vectoriel d'origine $\mathbb{R}^n$.

Exemple 4.2.1. Un point 3D $\mathbf{M} = (M_x, M_y, M_z) \in \mathbb{R}^3$ de l'espace 3D sera représenté dans l'espace projectif $\mathbb{P}^3$ par tous les vecteurs $(\lambda M_x, \lambda M_y, \lambda M_z, \lambda)$.

Les coordonnées homogènes permettent aussi de définir des points à l'infini, ou vecteurs. Néanmoins, dans le contexte de cette thèse, nous ne considérons que des points projectifs de dernière coordonnée non nulle.

4.2.2 Du modèle de la caméra à l'homographie

Nous rappelons ici le modèle de la caméra couramment utilisé en vision par ordinateur. Il s'agit du modèle sténopé [HZ04] (ou *pinhole*) qui modélise la caméra par une projection perspective.
Pour commencer, considérons la projection en trou d'épingle.

Soit un point $\mathbf{M} = (M_x, M_y, M_z) \in \mathbb{R}^3$ de l'espace, face à la caméra placée à l'origine du repère avec son plan image parallèle au plan XY et placé à une distance f de l'origine ; celui-ci est projeté dans le plan image au point $\mathbf{m}$ par la transformation $\mathbf{M} \mapsto \mathbf{m}$ suivante :

$$(M_x, M_y, M_z)^T \mapsto \left(\frac{fM_x}{M_z}, \frac{fM_y}{M_z} \right)^T . \tag{4.1}$$

En utilisant les coordonnées homogènes, cette projection perspective s'exprime comme une application linéaire, c'est à dire, un produit matriciel :

$$\begin{bmatrix} fM_x \\ fM_y \\ M_z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_z \\ 1 \end{bmatrix}, \tag{4.2}$$

où f désigne la distance focale du plan image. Le résultat sont les coordonnées homogènes du point $\mathbf{m}$ dans le plan image. En utilisant la propriété d'équivalence des points de l'espace projective, on peut réécrire l'équation 4.2

$$\begin{bmatrix} \frac{fM_x}{M_z} \\ \frac{fM_y}{M_z} \\ 1 \end{bmatrix} = \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_z \\ 1 \end{bmatrix} \quad \text{avec } \lambda = \frac{1}{M_z}. \tag{4.3}$$

On décompose ensuite l'équation 4.3 pour avoir :

$$\begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_z \\ 1 \end{bmatrix}. \tag{4.4}$$

On obtient finalement :

$$\mathbf{m} = \lambda \mathtt{K}_{3\times3} \left[\mathtt{I}_{3\times3} | \mathbf{0}_{3\times1} \right] \mathbf{M} = \mathtt{P}_{4\times3} \mathbf{M}. \tag{4.5}$$

Nous avons la relation entre un point de l'image et un point 3D de l'espace. Or le point $\mathbf{M}$ est actuellement exprimé dans le repère caméra. En général, les points de l'espace sont exprimés par rapport à un repéré "monde" arbitraire. Si on note $\mathbf{C}$ le centre de la caméra dans le repère monde, $\mathtt{R}$ la matrice de changement de base linéaire entre le repère monde et le repère caméra, $\mathbf{M}_{cam}$ un point 3D dans le référentiel caméra et $\mathbf{M}$ le même point 3D dans le référentiel monde, nous avons la relation suivante :

$$\mathbf{M}_{cam} = \mathtt{R}\,(\mathbf{M} - \mathbf{C}), \tag{4.6}$$

ou, de manière équivalente, sous forme matricielle

$$\mathbf{M}_{cam} = \begin{bmatrix} \mathtt{R} & -\mathtt{R}\,\mathbf{C} \\ 0 & 1 \end{bmatrix} \mathbf{M}. \tag{4.7}$$

L'équation 4.5 devient alors :

$$\mathbf{m} = \mathtt{P}\,\mathbf{M}_{cam} = \mathtt{K}\left[\mathtt{I}\,|\mathbf{0}\right]\mathbf{M}_{cam} = \mathtt{K}\left[\mathtt{I}\,|\mathbf{0}\right]\begin{bmatrix}\mathtt{R} & -\mathtt{R}\,\mathbf{C}\\ 0 & 1\end{bmatrix}\mathbf{M} = \mathtt{K}\left[\mathtt{R}\,|\mathbf{t}\right]\mathbf{M} = \mathtt{P}\,\mathbf{M} \quad (4.8)$$

La matrice $\mathtt{P}_{3\times4} = \mathtt{K}_{3\times3}\left[\mathtt{R}_{3\times3}|\mathbf{t}_{3\times1}\right]$ est appelée la **matrice de projection**. La matrice $\left[\mathtt{R}_{3\times3}|\mathbf{t}_{3\times1}\right]$, quant à elle, est appelée la matrice des **paramètres extrinsèques** de la caméra et la matrice $\mathtt{K}_{3\times3}$ est appelée la matrice des **paramètres intrinsèques**.

Nous nous intéressons maintenant à la matrice $\mathtt{K}$ car il faut considérer d'autres éléments.
Actuellement, l'origine du repère dans le plan-image est le centre de l'image. Comme le traitement des images se fait généralement avec un repère dont l'origine se trouve dans un des coins (souvent en haut, à gauche), une translation du repère est effectuée. L'équation 4.4 devient donc :

$$\begin{bmatrix}m_x\\ m_y\\ 1\end{bmatrix} = \lambda\begin{bmatrix}f & & p_x\\ & f & p_y\\ & & 1\end{bmatrix}\begin{bmatrix}1 & & & 0\\ & 1 & & 0\\ & & 1 & 0\end{bmatrix}\begin{bmatrix}M_x\\ M_y\\ M_z\\ 1\end{bmatrix}, \quad (4.9)$$

avec $\mathbf{p} = (p_x, p_y)$ le vecteur de translation entre le centre de l'image est le coin de l'image considéré comme origine du repère image.

Ensuite, les caméras CCD n'ont pas une représentation en pixel, il faut donc faire une association entre les éléments CCD et les pixels. Les grandeurs k_x et k_y sont introduites pour effectuer la conversion : k_x (respectivement k_y) désigne le nombre de pixels par unité de longueur suivant la direction x (respectivement y). La matrice $\mathtt{K}$ de l'équation 4.9 devient ainsi :

$$\mathtt{K} = \begin{bmatrix}k_x & & \\ & k_y & \\ & & 1\end{bmatrix}\begin{bmatrix}f & & p_x\\ & f & p_y\\ & & 1\end{bmatrix} = \begin{bmatrix}f_x & & c_x\\ & f_y & c_y\\ & & 1\end{bmatrix}, \quad (4.10)$$

où c_x et c_y (en pixels) désignent les coordonnées de l'intersection de l'axe optique avec le plan image (théoriquement au centre de l'image) et $f_x = fk_x$ et $f_y = fk_y$ désignent la focale de la caméra en nombre de pixels suivant les directions x et y respectivement. Comme les pixels sont supposés carrés, nous posons $k_x = k_y$
Enfin, pour compléter le modèle de projection et tenir en compte de la non orthogonalité éventuelle des lignes et des colonnes, on peut introduire dans l'équation 4.10 un paramètre θ (appelé *skew factor*) :

$$\mathtt{K} = \begin{bmatrix}f_x & f_x\cot\theta & c_x + c_y\cot\theta\\ 0 & \frac{f_y}{\sin\theta} & \frac{c_y}{\sin\theta}\\ 0 & 0 & 1\end{bmatrix}. \quad (4.11)$$

Cependant, dans les caméras CCD modernes, le procédé de fabrication est tel que on peut raisonnablement assumer $\theta = \frac{\pi}{2}$. On retiendra pour $\mathtt{K}$ l'équation 4.10 plutôt que l'équation 4.11. Au final, $\mathtt{K}$ est appelée la matrice des **paramètres intrinsèques** de la caméra. Pour résumer, la figure 4.1 rappelle les trois transformations élémentaires successives pour passer d'un point 3D à un point-image :

— la transformation entre le repère monde et celui de la caméra ((1) de la figure 4.1),
— la transformation entre le repère de la caméra et le repère capteur (plan rétinien) ((2) de la figure 4.1),

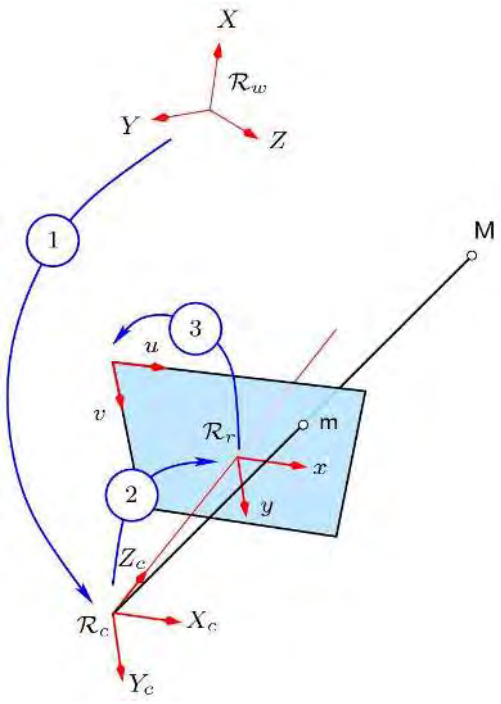


FIGURE 4.1 – Modèle sténopé de la caméra (figure appartenant au domaine public et issue de http://www.optique-ingenieur.org) (1) représente la transformation entre le repère monde $\mathcal{R}_w$ et celui de la caméra $\mathcal{R}_c$. (2) représente la transformation entre le repère de la caméra $\mathcal{R}_c$ et le repère capteur (plan rétinien) $\mathcal{R}_r$. (3) représente la transformation entre le repère capteur $\mathcal{R}_r$ et le repère image.

— la transformation entre le repère capteur et le repère image ((3) de la figure 4.1).

Maintenant, considérons le cas particulier où les points $\mathbf{M}$ appartiennent à un même plan Δ. De façon arbitraire, on peut placer le repère monde de sorte que les points de ce plan vérifient $M_z = 0$. Nous appliquons l'équation 4.8 à ce cas particulier :

$$\begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} = \lambda \mathtt{K} \begin{bmatrix} \mathtt{R} | \mathbf{t} \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ 0 \\ 1 \end{bmatrix}. \tag{4.12}$$

Nous simplifions l'équation 4.11 pour ne plus tenir compte de la coordonnée en z :

$$\begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} = \lambda \mathtt{K}\, \mathtt{R} \begin{bmatrix} 1 & 0 & \\ 0 & 1 & \mathbf{t} \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ 1 \end{bmatrix} = \lambda \mathtt{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ 1 \end{bmatrix} = \lambda \mathtt{H} \begin{bmatrix} M_x \\ M_y \\ 1 \end{bmatrix} \tag{4.13}$$

où $\mathtt{R} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix}$. La matrice de projection $\mathtt{H}_{3\times3} = \lambda \mathtt{K}_{3\times3} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}_{3\times3}$ dans ce cas particulier est appelée l'**homographie** entre le plan Δ et le plan image de la caméra.

4.2.3 Utilisations

La formulation de l'homographie $\mathtt{H}$ de l'équation 4.13 montre que l'homographie dépend des paramètres de pose de la caméra. Si la caméra est calibrée ($\mathtt{K}$ connue) et si nous avons calculé l'homographie $\mathtt{H}$, nous pouvons facilement retrouver la pose

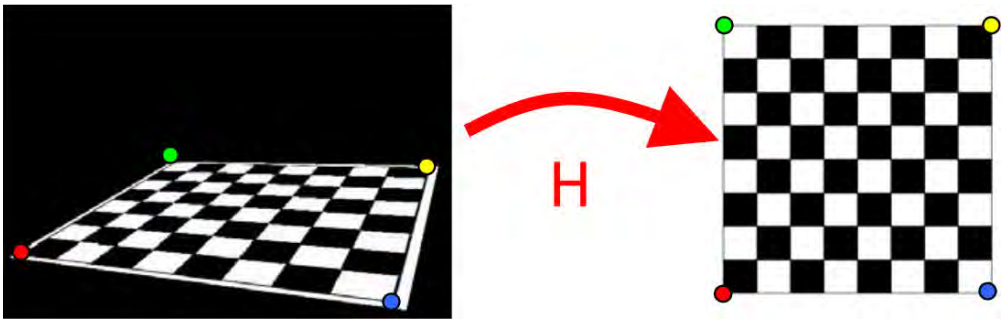


FIGURE 4.2 – Calcul de l'homographie **H** entre les points du plan et leur projection dans l'image d'une caméra (figure extraite d'une présentation de Christiano Gava).

de la caméra (c'est-à-dire `R` et $\mathbf{t}$) par décomposition de `H` [MV07]. Si les projection de points du plan Δ sont connues dans le plan image, on peut estimer `H` : au moins 4 correspondances entre les points du plan et leurs points associés sur le plan image sont nécessaires [HZ04]. Les applications de Réalité Augmentée utilisent souvent des marqueurs dont la structure est connue. Cela permet d'estimer facilement `H` et donc d'en déduire la pose de la caméra. Dans l'exemple de la figure 4.2, les 4 coins du marqueur en forme d'échiquier sont connus (car on connaît la taille du marqueur). Ainsi, `H` peut être calculée à partir des correspondances entre les 4 coins et leurs points sur le plan image de droite.

L'homographie `H` étant la transformation entre les points (en 3D) du plan Δ et des points de l'image (donc dans le plan image), elle permet donc de passer d'un espace planaire à l'autre. En outre, l'homographie élimine la distorsion perspective de l'image d'un plan pour obtenir une image "rectifiée" du plan. Dans notre contexte de Réalité Diminuée, l'objectif de la rectification est de trouver l'homographie `H` qui, à un facteur d'échelle près, transforme une image définie sur un domaine du plan caméra en une image définie sur un domaine d'un plan parallèle à un plan de la pièce d'intérieur (sol ou mur).

La figure 4.3 montre les étapes de la rectification par un rapport à un plan Δ. À partir d'une image I prise par une caméra C, les pixels sont transformés par l'homographie calculée `H` (*cf*. l'équation 4.13). Une fois la rectification effectuée, nous disposons d'une nouvelle image appelée image *rectifiée*. Nous utilisons ensuite notre méthode d'*inpainting* du chapitre 3 pour compléter la zone masquée rectifiée Ω_r ainsi que les zones non visibles par rapport à la caméra C (zone en rouge dans la figure 4.3). Une fois l'*inpainting*, il suffit d'appliquer $\texttt{H}^{-1}$ à l'image rectifiée pour retrouver les valeurs du point de vue d'origine.

La rectification permet d'envoyer les points et les lignes de fuite à l'infini et d'enlever la perspective de la texture du plan considérée (les lignes appartenant au même plan Δ, parallèles en 3D dans la scène, sont parallèles dans l'image rectifiée). L'avantage est de disposer ainsi d'une image où la texture peut être considérée comme étant vue de face. La complétion par l'*inpainting* s'en retrouve facilitée car les motifs de la texture et leur distribution sont réguliers dans l'image rectifiée (voir figure 4.4). Il suffit ensuite d'appliquer l'homographie inverse à l'image rectifiée complétée et de remplacer les valeurs de la zone masquée Ω par celles complétées dans l'image rectifiée.

Cependant, la rectification implique d'interpoler les valeurs des pixels où l'information est éparse car trop peu détaillée dans l'image d'origine. Dans la prochaine section, nous allons caractériser ce problème de résolution et ses conséquences dans le cadre de l'*inpainting*.

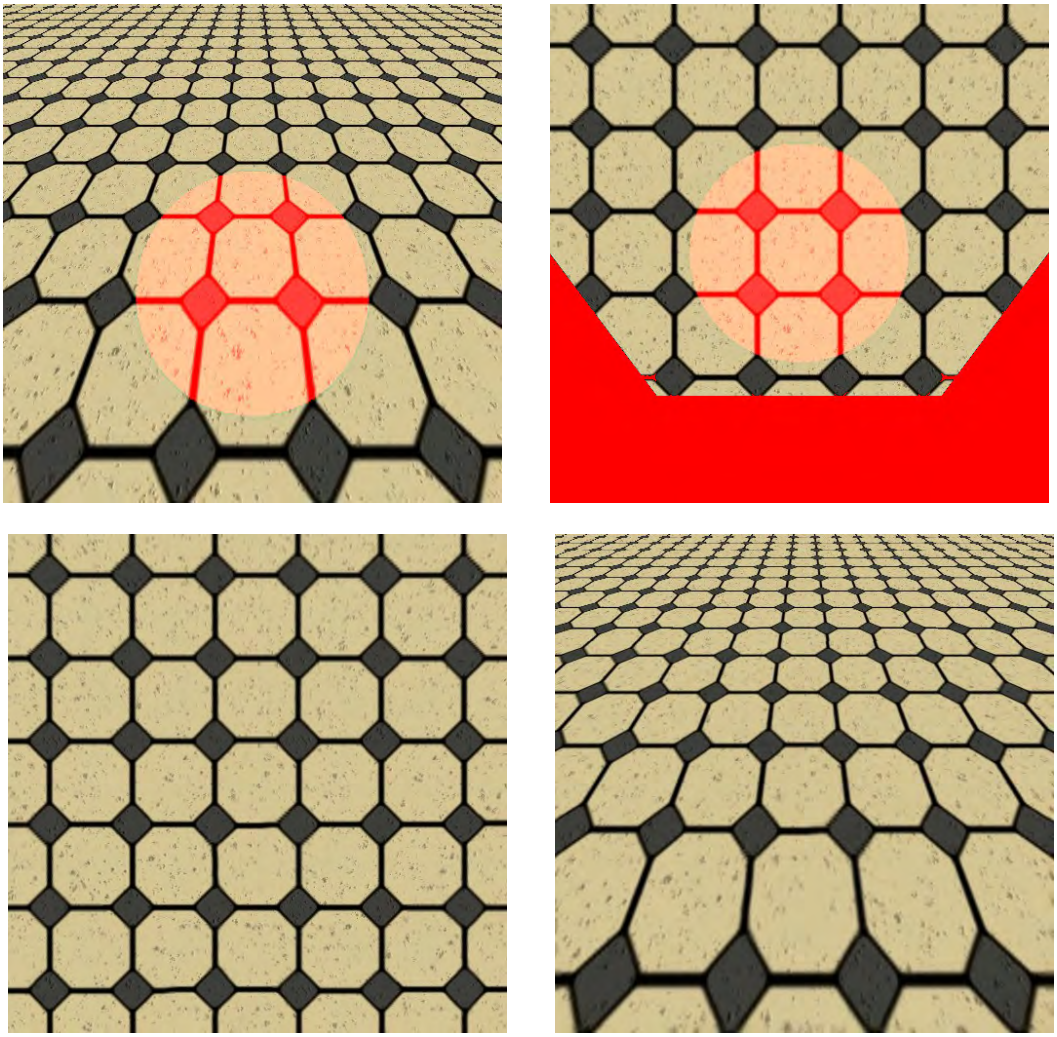


FIGURE 4.3 – Étape de rectification d'une texture. À partir de l'image de la prise de vue (en haut, à gauche) et de l'homographie calculée, on en obtient une image rectifiée (en haut, à droite) où la déformation due à la perspective a été enlevée. La zone masquée (cercle colorié en rouge) ainsi que la zone inconnue coloriée en rouge (une zone où nous n'avons pas d'information depuis le point de vue considéré) sont ensuite complétées par une approche d'*inpainting* (en bas, à gauche). L'image complétée est de nouveau projetée dans l'espace d'origine via l'homographie inverse (en bas, à droite).

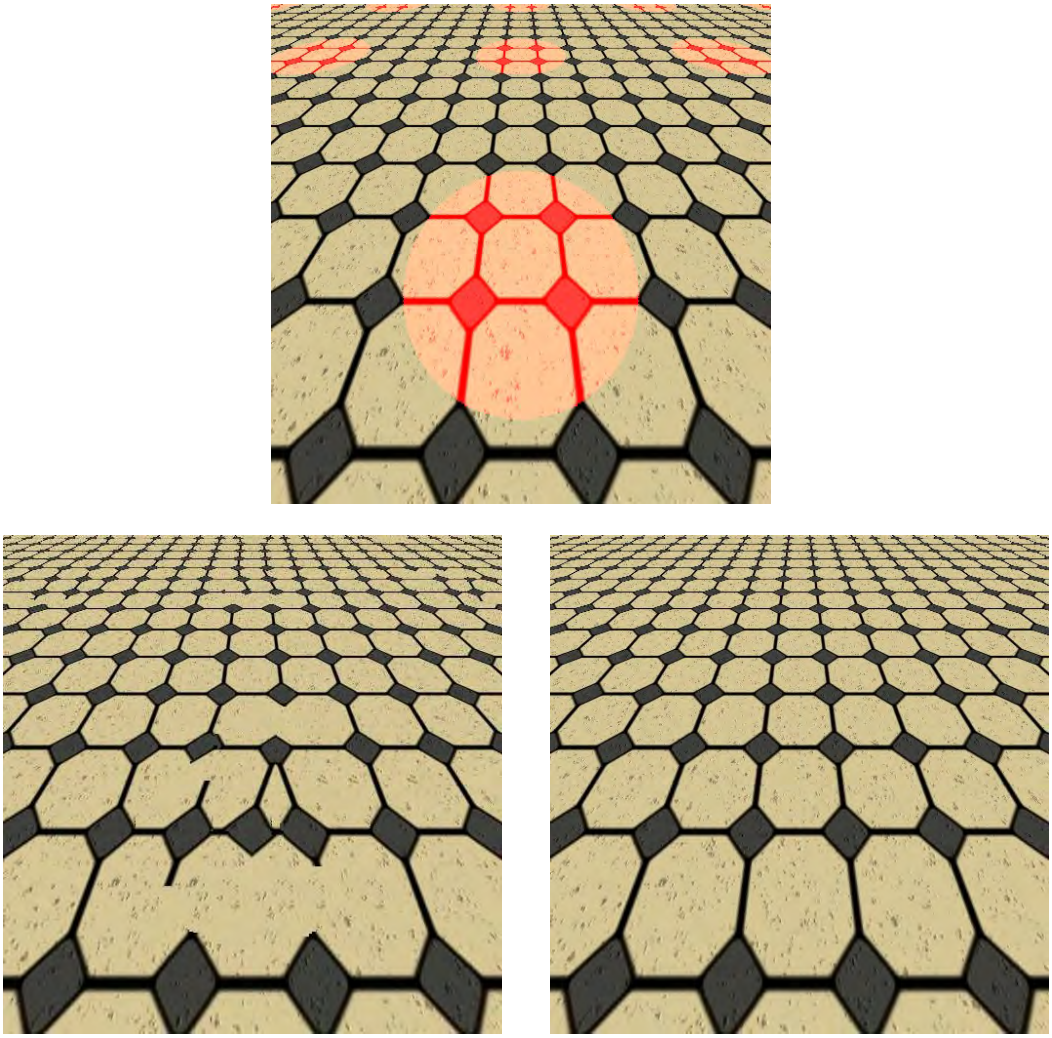


FIGURE 4.4 – Résultat de complétion par une approche basée *offsets* sur une texture régulière (première ligne). Sur la deuxième ligne, l'image de gauche est complétée sans rectification tandis que l'image de droite est complétée avec rectification. Nous pouvons constater que l'image complétée, où aucune rectification n'a été faite, a un résultat qui ne respecte pas la structure de la texture. À l'inverse, la structure a été respectée dans l'image où la rectification a été effectuée.

4.3 Le problème de la résolution

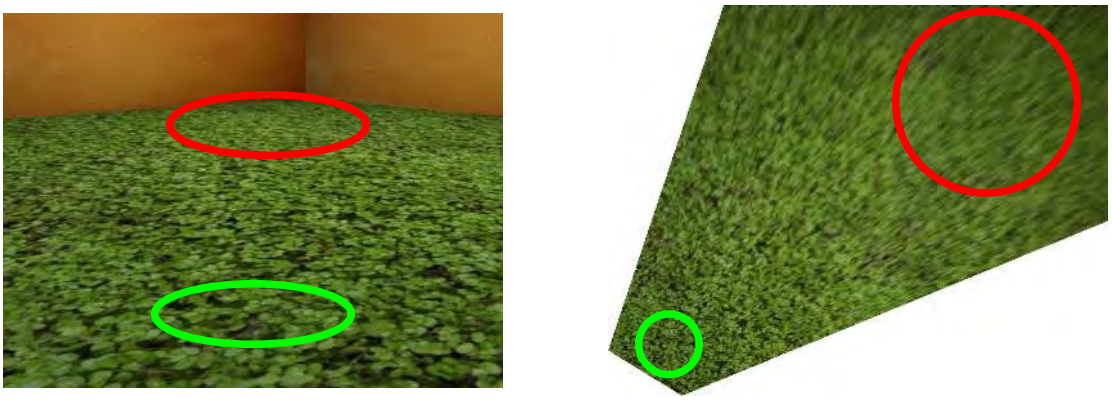


FIGURE 4.5 – Rectification d'une image par rapport au plan du sol. Les deux zones entourés dans l'image de gauche ont une aire différente dans l'image rectifiée de droite. Comme la zone verte est plus proche de la caméra, sa zone projetée sera plus petite que la zone rouge située plus loin de la caméra.

Dans notre processus de Réalité Diminuée, nous appliquons une approche d'*inpainting* à une image rectifiée. Celle-ci a sa structure qui n'est plus déformée à cause de la perspective. Cependant, cette vue non déformée ne possède pas une qualité d'information uniforme. En effet, considérons une prise de vue où l'on voit le plan du sol comme dans la figure 4.5. Nous avons des données qui sont détaillées comme la texture en bas de l'image (entourée en vert) car cette partie de la texture est proche de la caméra. À l'inverse, la partie de la texture située à la frontière avec les deux autres plans (entourée en rouge) est beaucoup moins détaillée car éloignée du point de vue.

En appliquant l'homographie qui permet de rectifier l'image, deux *patchs* de même taille (l'un venant de la zone verte, l'autre de la zone rouge) auront leur projection dans l'image rectifiée qui aura une surface différente (la surface de la projection du *patch* de la zone rouge sera bien plus grande que celle du *patch* de la zone verte). Or dans l'image rendue de la scène, chacune de de ces zones contient la même quantité d'information. Du coup, sur l'image rectifiée, l'information dans le cercle rouge sera bien plus éparse que dans le cercle vert. Au final, la zone verte sera très détaillée alors que la zone rouge sera plus floue. Nous allons voir maintenant comment cette différence de résolution a une influence sur le processus d'*inpainting*. Pour cela, considérons un plan texturé Π vu par une caméra C selon un angle γ par rapport à la normale $\overrightarrow{n}$ du plan comme schématisé dans la figure 4.6.

Nous générons plusieurs points de vue en faisant varier l'angle γ. Cela va accentuer l'angle d'inclinaison de la caméra par rapport à la normale du plan. Nous notons I_γ l'image représentant une prise de vue selon l'angle γ. La figure 4.7 montre différentes images I_γ (colonne (a)) en fonction de l'angle γ. La zone encadrée en rouge correspond au masque d'effacement. Chaque image est rectifiée selon l'homographie $\mathtt{H}_\gamma$ associée (colonne (b)). Ensuite, la zone masquée est complétée par un procédé d'*inpainting* (ici *PatchMatch*). Nous constatons que la zone de l'image où l'information est la plus éparse s'est propagée dans toute la zone masquée, créant des différences de résolution dans la zone complétée ; ces différences sont d'autant plus visibles et gênants que l'angle γ est élevé (colonne (c)). L'effet est beaucoup plus visible une fois l'homographie inverse $\mathtt{H}_\gamma^{-1}$ appliquée. En effet, on constate une zone floue dont l'intensité est fonction de l'angle γ dans la zone masquée (colonne (d)).

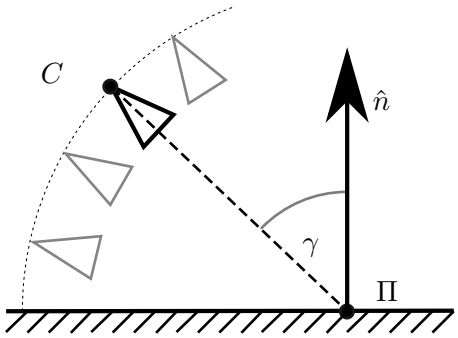


FIGURE 4.6 – Position de la caméra par rapport au plan Π et à l'angle γ.

Cela est dû au fait que les approches comme *PatchMatch* sont des approches multi-résolutions. Les grandes lignes de la carte de correspondance (à savoir, où chercher l'information dans les résolutions supérieures) sont dessinées en basse résolution. Or, à ce niveau de résolution, la qualité des données éparses est équivalente à celle des données détaillées. L'approche *PatchMatch* va donc sélectionner des pixels sans identifier cette différence de détail, uniquement visible en haute résolution. De plus, comme la zone des données avec peu de détail a une surface bien plus grande que celle avec beaucoup de détail, la probabilité de sélectionner un pixel représentant une information peu détaillée est donc très élevée d'où une propagation de ces données.

Dans la prochaine section, nous introduisons un critère permettant de contrôler cette propagation.

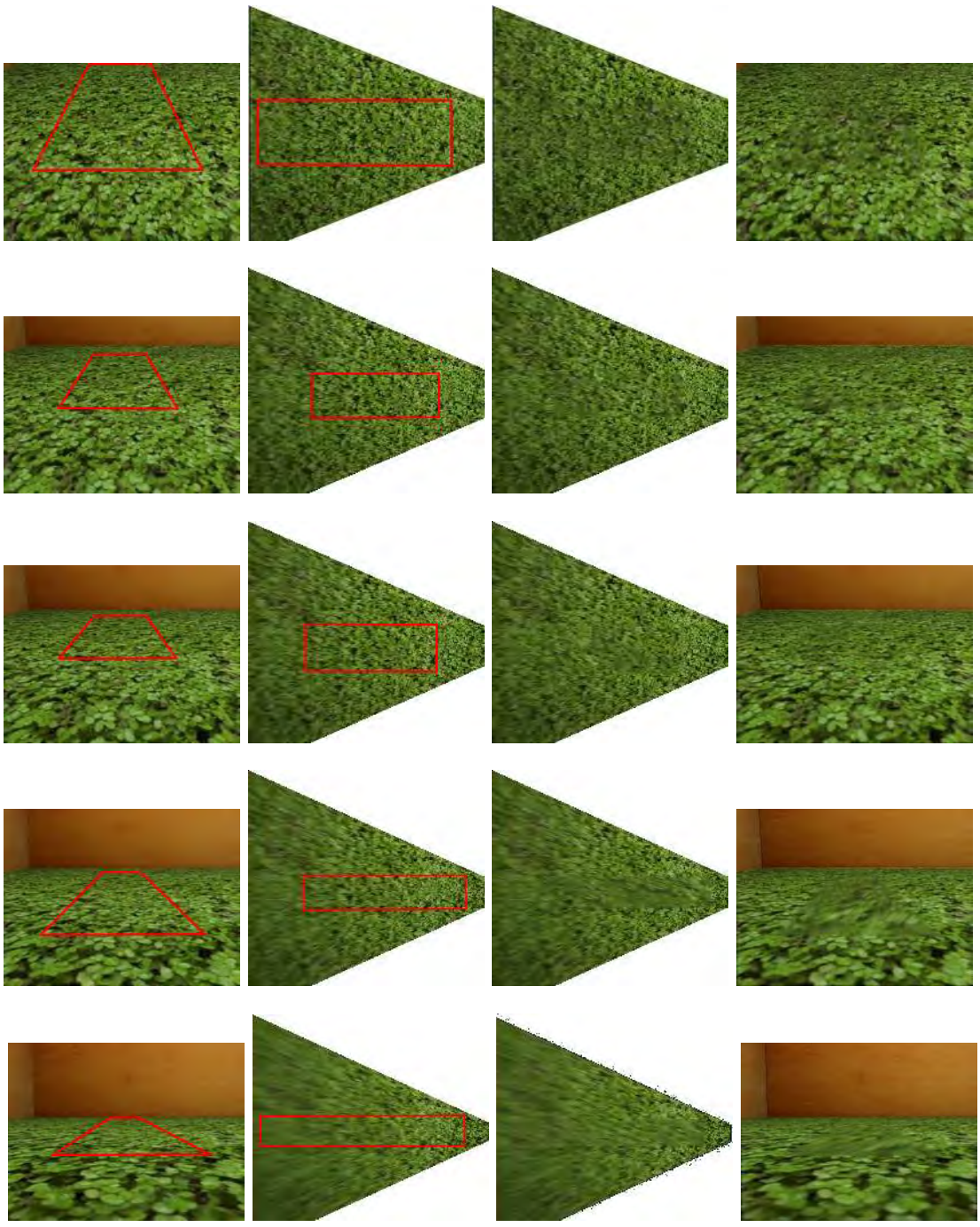


FIGURE 4.7 – Influence de la perspective sur le processus d'*inpainting* sur plusieurs images de synthèse générées avec, en descendant de ligne en ligne, un angle γ égal à 60°, 70°, 75°, 80° et 85°. De gauche à droite : (a) l'image d'entrée avec la zone encadrée qui est effacée par l'*inpainting*, (b) l'image rectifiée du plan du sol, (c) l'image rectifiée complétée par un *inpainting* classique, (d) l'image de sortie complétée.

4.4 Critère de confiance radiométrique

Dans cette section nous explicitons notre critère de confiance calculé sur l'image rectifiée. Nous cherchons à associer à un chaque pixel un score caractérisant la qualité de sa projection lorsqu'il est transformé par l'homographie vers l'image rectifiée. Après avoir énoncé quelques éléments de notation, nous présentons la loi de Bouguer ainsi que le cheminement qui conduit à notre critère de confiance.

Éléments de notation

Nous rappelons d'abord la notion d'angle solide qui est l'analogue tridimensionnel de la notion d'angle dans un plan.

Définition 4.4.1. Soit une sphère $\mathcal{S}$ de rayon R et de centre $\mathbf{A}$. L'*angle solide* d'un cône de sommet A est numériquement égal au rapport entre l'aire $\mathcal{A}_c$ de la calotte sphérique $\mathcal{S}_c$ découpée par ce cône sur la sphère $\mathcal{S}$ et du rayon R de $\mathcal{S}$ élevé au carré :

$$\Omega = \frac{\mathcal{A}_c}{R^2} \tag{4.14}$$

L'unité est le stéradian (sr).

Nous donnons également la définition d'un angle solide élémentaire de sommet $\mathbf{A}$ par rapport à une surface infinitésimale centrée en un point $\mathbf{M}$:

$$\mathrm{d}\,\Omega = \frac{\vec{\mathbf{u}} \cdot \vec{\mathbf{n}}}{r^2}\,\mathrm{d}\,S \tag{4.15}$$

où $\mathrm{d}\,\Omega$ est l'angle solide élémentaire, $\mathrm{d}\,S$ est la surface élémentaire et $\vec{\mathbf{n}}$ sa normale, $\vec{\mathbf{u}}$ est le vecteur unitaire porté par $\overrightarrow{\mathbf{AM}}$ et r est la distance séparant le sommet de l'angle solide élémentaire A et l'élément de surface $\mathrm{d}\,S$.

Nous donnons quelques définitions spécifiques à la radiométrie.

Définition 4.4.2. Un *flux énergétique* (noté F) est une quantité d'énergie par unité de temps, exprimé en Watt (W).

Le flux énergétique est une puissance. Elle ne donne aucune information sur la distribution angulaire du rayonnement ou la géométrie de la source. Deux autres grandeurs sont alors introduites : l'éclairement énergétique et l'intensité énergétique.

Définition 4.4.3. L'*éclairement énergétique* (noté E) est une densité de flux énergétique reçu par unité de surface :

$$E = \frac{\mathrm{d}\,F}{\mathrm{d}\,S} \tag{4.16}$$

Il s'exprime en $\mathrm{W\,m^{-2}}$

Définition 4.4.4. L'*intensité énergétique* (notée I) est le flux émis par unité d'angle solide dans une direction d'observation donnée, exprimé en $\mathrm{W\,sr^{-2}}$. Sauf pour des cas particuliers (étoiles, sources lambertiennes), l'intensité d'une source est en général non isotrope, c'est à dire qu'elle dépend en général de la direction d'observation.

Loi de Bouguer

Nous énonçons et analysons dans cette sous-section la loi de Bouguer.

Énoncé

Notre critère repose sur la loi de Bouguer qui mesure le flux lumineux $\mathrm{d}\,F$ d'une source lumineuse d'intensité I à travers une surface $\mathrm{d}\,S$ vu sous un angle solide $\mathrm{d}\,\Omega$. L'équation de cette loi est la suivante :

$$\mathrm{d}\,F = I\,\mathrm{d}\,\Omega = \frac{\vec{\mathbf{u}} \cdot \vec{\mathbf{n}}}{r^2}\,\mathrm{d}\,S = I\frac{\cos\theta}{r^2}\,\mathrm{d}\,S. \tag{4.17}$$

où r est la distance entre le centre de la surface et la source lumineuse, θ est l'angle entre le rayon lumineux et la normale $\vec{n}$ de la surface $\mathrm{d}\,S$. La figure 4.8 schématise l'application de cette loi.

Avant d'appliquer cette loi dans notre problème de vision par ordinateur, nous expliquons dans la prochaine sous-section ce qu'elle signifie intuitivement.

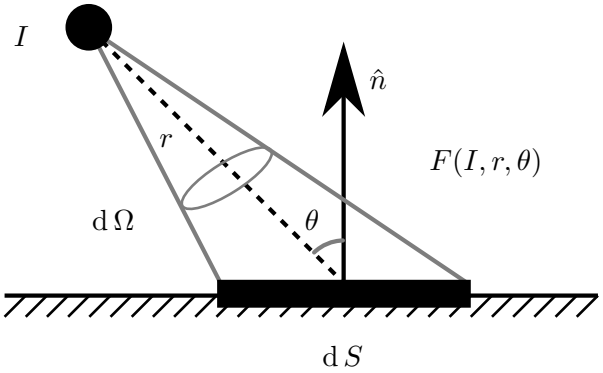


FIGURE 4.8 – Application de la loi de Bouguer pour une source lumineuse d'intensité I sur une surface $\mathrm{d}\,S$.

Analyse

Nous nous permettons de faire quelques analogies. Nous avons l'intensité lumineuse qui est une grandeur caractérisant une source ponctuelle selon une direction donnée. Elle n'est également pas évaluable du point de vue d'un récepteur. À l'inverse, le flux caractérise la puissance lumineuse, telle qu'elle est perçue par le récepteur (par exemple l'œil humain). Le flux est conservatif ce qui signifie qu'il est constant pour un même angle solide. Cela se vérifie d'ailleurs dans l'équation 4.17 en divisant par $\mathrm{d}\,\Omega$:

$$\frac{\mathrm{d}\,F}{\mathrm{d}\,\Omega} = I \tag{4.18}$$

et car I est constant selon une direction. Par exemple dans l'image de gauche de la figure 4.9, le flux reçu sur les surface S_1 et S_2 est le même car représentant le même cône généré par l'angle solide Ω.

Cependant, si on prend une surface de taille fixe $\mathrm{d}\,S$ et qu'on la déplace dans l'angle solide Ω, le flux reçu sur cette surface va varier en fonction de plusieurs variables. Les variables sont la distance et l'angle entre la direction du rayon lumineux et la normale au plan tangent à la surface. En considérant cette surface comme infinitésimale, nous arrivons à la grandeur de l'éclairement. Celle-ci correspond à la sensation en cette surface de la puissance lumineuse reçue.

Nous prenons maintenant deux exemples de la vie courante. Le premier exemple illustre la variation du flux pour une surface donnée en fonction de la distance. Nous considérons une ampoule près de laquelle nous plaçons notre main à 1 cm, la paume

face à l'ampoule. Nous déplaçons notre main dans la direction d'un rayon lumineux en l'éloignant de l'ampoule. Nous avons une sensation de chaleur plus élevée lorsque notre main se trouve à 1 cm que lorsqu'elle est à 5 cm. Cela correspond à la situation des éléments de surface de l'image de gauche figure 4.9 où $\mathrm{d}\,S_1$ va recevoir un flux plus élevée que $\mathrm{d}\,S_2$. Le deuxième exemple, quant à lui, illustre l'influence de l'angle entre la normale de la surface et le rayon lumineux (voir la deuxième image de la figure 4.9). On considère deux personnes présentes sur la surface de la Terre, en exagérant un peu l'angle de l'axe de rotation de la Terre, durant la période du mois de décembre, l'un étant dans l'hémisphère Sud et l'autre dans l'hémisphère Nord. Ces deux personnes agissent comme deux éléments de surface dont leurs normales correspondent à la position debout. De plus, la distance séparant les deux personnes est négligeable devant la distance entre la Terre et le Soleil. Nous pouvons supposer que les deux personnes sont à la même distance du Soleil. Pourtant, la première personne aura une sensation de chaleur plus élevée alors que la deuxième aura plutôt une sensation de froid. Cela s'explique par l'influence de l'angle entre la normale de la surface et le rayon lumineux dans la loi de Bouguer.

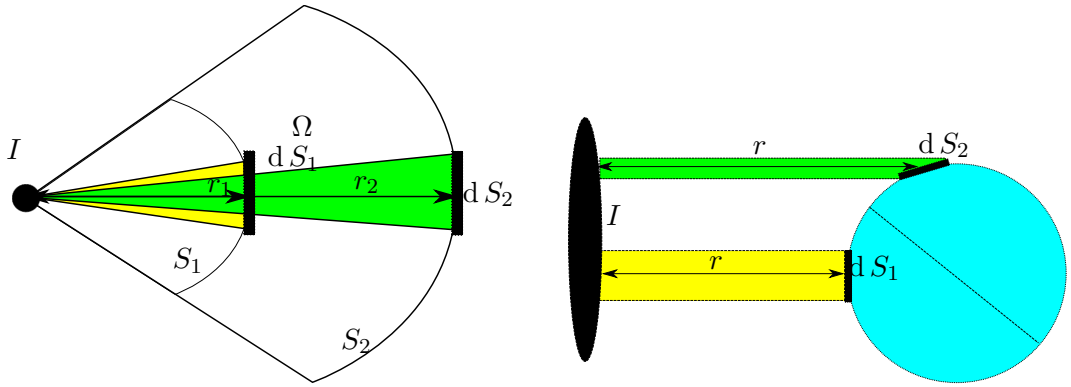


FIGURE 4.9 – Loi de Bouguer : influence de la distance et de l'angle dans le calcul du flux. Dans les deux exemples, la surface $\mathrm{d}\,S_2$ recevant le flux coloré en vert a un éclairement plus faible que la surface $\mathrm{d}\,S_1$ recevant le flux coloré en jaune.

Dans les deux cas, la sensation de chaleur correspond à l'intensité du flux surfacique $\frac{\mathrm{d}\,F}{\mathrm{d}\,S}$. Plus cette grandeur est élevée, plus la sensation de chaleur l'est également.

4.4.1 Application à la vision par ordinateur

Maintenant que nous avons expliqué la notion de flux à travers la loi de Bouguer, nous allons montrer que cette quantité est appropriée pour notre critère de confiance. Nous partons, pour cela, de l'équation 4.17 Nous considérons une caméra C et un plan Π arbitraire. Nous considérons C comme une source lumineuse avec une intensité lumineuse I (voir figure 4.10).

L'éclairement (à savoir le flux lumineux selon un élément de surface) est donc :

$$E = \frac{\mathrm{d}\,F}{\mathrm{d}\,S} = I\frac{\cos\theta}{r^2}. \tag{4.19}$$

Nous exprimons l'éclairement, en chaque point $\mathbf{P}$ du plan image, ainsi que l'éclairement de son projeté $\mathbf{Q}$ sur le plan Π en utilisant équation 4.19 ce qui donne :

$$E(\mathbf{P}) = I\frac{\cos\phi}{r_P^2} = I\frac{\cos^3\phi}{f^2} \tag{4.20}$$

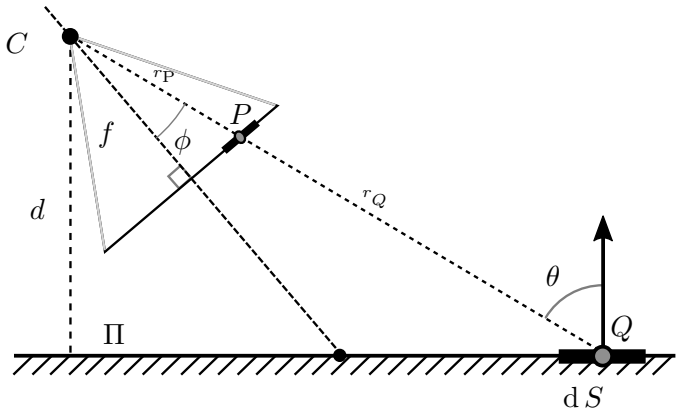


FIGURE 4.10 – Transposition de la loi de Bouguer dans le cas de la vision par ordinateur. Le points **Q** est la projection du point **P** du plan image sur le plan Π et sa surface associée d S.

où ϕ est l'angle entre le rayon lumineux (**CP**) et l'axe de la caméra C.

$$E(\mathbf{Q}) = I\frac{\cos\theta}{r_Q^2} = I\frac{\cos^3\theta}{d^2}, \tag{4.21}$$

où f est la distance focale de la caméra C et d est la distance entre C et Π. Notons que :

$$r_P = \frac{f}{\cos\phi}, \quad r_Q = \frac{d}{\cos\theta} \tag{4.22}$$

Nous définissons notre critère nommé trust comme le ratio entre l'éclairement du point **Q** et l'éclairement du point **P** (en utilisant (4.20) et (4.21)) :

$$\text{trust}(\mathbf{P}) = \frac{E(\mathbf{Q})}{E(\mathbf{P})} = \left(\frac{f}{d}\right)^2 \left(\frac{\cos\theta}{\cos\phi}\right)^3. \tag{4.23}$$

Notons que les valeurs diminuent pour une distance d croissante et pour un angle θ croissant. Les pixels qui représente des régions du plan Π proches de la caméra et vue sous un angle presque nul vont avoir une valeur élevée par la fonction trust alors que les pixels qui représentent des régions du plan Π éloignées de la caméra et/ou vues sous un angle très prononcé vont avoir une valeur plus faible par la fonction trust. Les pixels ayant une qualité élevé auront donc une valeur de confiance élevée alors que les pixels ayant une qualité basse auront une valeur de confiance faible

Carte de confiance

Nous avons définit notre critère de confiance pour un point **P** de l'espace. Nous devons maintenant caractériser le critère pour un pixel p de l'image rectifiée. Nous considérons **P** comme le centre du pixel p de l'image. Nous définissons la *carte de confiance* $\mathcal{C}$ pour chaque pixel du domaine $U \subset \mathbb{N}^2$ de l'image rectifiée

$$\mathcal{C} : \begin{cases} U & \longrightarrow \mathbb{R} \\ \mathbf{p} & \longmapsto \mathcal{C}(\mathbf{p}) = \text{trust}(\mathbf{P}) \end{cases} \tag{4.24}$$

avec $\mathcal{C}(\mathbf{p}) = 0$ si **p** n'est pas une projection d'un point du plan caméra. La figure 4.11 montre un exemple de la carte appliquée à une image de synthèse.

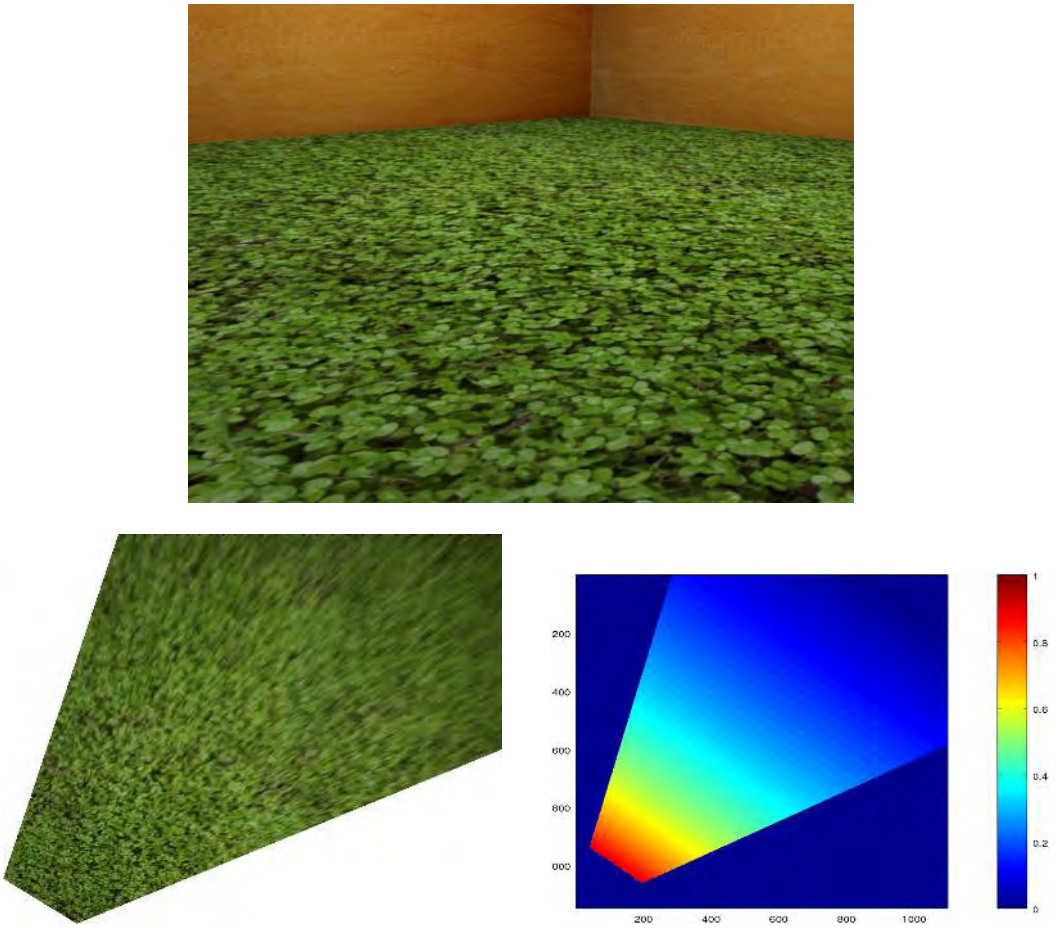


FIGURE 4.11 – Carte de confiance d'une image. Gauche : image d'entrée d'un plan du sol (texture verte). Centre : image rectifiée du plan. Droite : carte de confiance $\mathcal{C}$ de l'image rectifiée, le code couleur (valeurs élevées e, rouge et valeurs basses en bleu) montre que la valeur de trust pour les pixels qui ont une qualité d'information basse, à savoir ceux situés en haut à droite de l'image.

Dans la prochaine section, nous montrons comment nous intégrons notre critère dans les deux méthodes d'*inpainting* analysées dans le chapitre 2. Notons que notre critère n'est pas restreint à ces méthodes et qu'il peut être appliqué à d'autres applications comme le filtrage ou dans d'autres approches d'*inpainting*.

4.5 Applications à l'*inpainting*

Dans cette section, nous abordons les modifications effectués dans l'approche d'*inpainting* basée sur *PatchMatch* [Bar+09] et l'approche basée sur l'analyse statistique des *offsets* [HS12b], et par extension, à notre approche du chapitre 3. Nous y ajoutons une fonction, dite de validation, qui va contrôler la propagation des pixels ayant une confiance faible. Nous y détaillons également une dernière application de ce critère, à savoir un flou gaussien dont l'intensité varie en fonction dudit critère afin de réduire les discontinuités créées après la copie des informations de qualité haute dans les régions de qualité moindre.

4.5.1 Fonction de validation

Premièrement, nous proposons de définir une fonction prédicat qui sera utilisée dans les deux approches d'*inpainting* énoncées ci-dessus. Cette fonction permet de gérer la sélection d'un pixel candidat en utilisant la carte de confiance $\mathcal{C}$. Cela permet d'éviter la sélection de données de « basse qualité » :

$$\text{validation}(\mathbf{p_1}, \mathbf{p_2}, \mathcal{C}, \alpha) = \mathcal{C}(\mathbf{p_2}) \leqslant \alpha\, \mathcal{C}(\mathbf{p_1}), \tag{4.25}$$

où $\mathbf{p_1}$ est le pixel à compléter, $\mathbf{p_2}$ est le candidat proposé par l'approche et $\alpha \in]0, 1]$ est un paramètre de tolérance.

4.5.2 Prise en compte de la résolution dans *PatchMatch*

Comme expliqué dans le chapitre 2, *PatchMatch* [Bar+09] cherche efficacement des correspondances entre les *patchs* de deux images ou entre la zone masquée et la zone connue dans le cadre de l'*inpainting* selon une fonction de similarité. La carte est initialisée de manière aléatoire ou via une analyse d'*offsets* comme dans notre approche du chapitre 3. Elle est ensuite affinée avec une pyramide multi-résolutions et en effectuant deux mises à jour : (a) les correspondances les plus pertinentes sont propagées aux pixels voisins, et (b) une recherche aléatoire autour du pixel candidat est effectuée pour trouver un pixel avec un meilleur score.

Dans *PatchMatch*, les *patchs* candidats sont équiprobables, indépendamment de la qualité de la rectification, particulièrement en basse résolution. Cela conduit à une propagation des données de mauvaise qualité à travers le masque.
Pour éviter la propagation de données de faible résolution, nous modifions la phase de propagation ainsi que dans la phase de randomisation (texte écrit en rouge dans l'algorithme 5). On vérifie, pour un *patch* à compléter de Ω, qu'un *patch* candidat a un meilleur score de similarité que le *patch* assigné courant dans la carte de correspondance ; nous nous assurons aussi que le pixel central $\mathbf{q}$ du *patch* candidat a une meilleure confiance que le pixel central $\mathbf{p}$ du *patch* à compléter. Pour cela, nous utilisons la fonction de validation en prenant $\alpha = 1$ dans ce cas.
La figure 4.12 illustre visuellement les effets de cette modification. On y voit la carte de correspondance obtenue après l'*inpainting* par *PatchMatch*. Nous avons utilisé le système de couleur de sorte que les données de forte qualité apparaissent vertes tandis que les données de faible qualité apparaissent rouges. La couleur à l'intérieur du masque Ω correspond à l'endroit où l'information a été copiée. Nous constatons que dans l'image de gauche, beaucoup de données labellisées en rouge ont été copiés dans une zone où l'on attend une donnée de forte qualité (en bas, à gauche). À l'inverse, dans l'image de droite, ce sont plutôt les données labellisées en vert qui ont été copiées dans la zone du

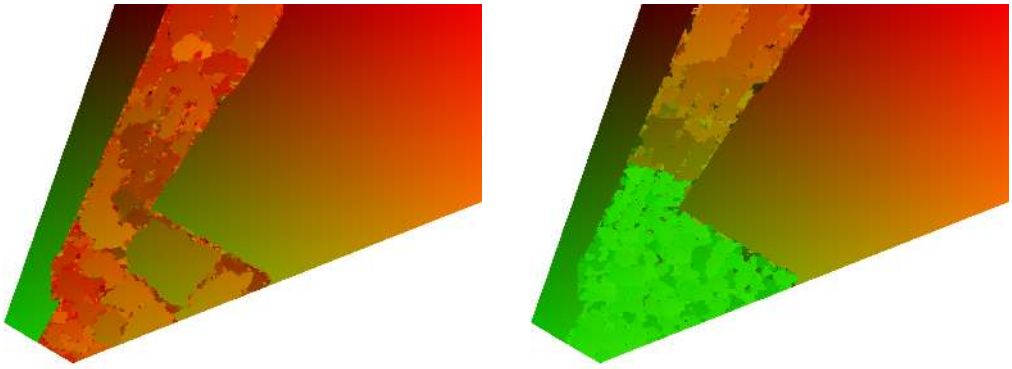


FIGURE 4.12 – Les effets de nos modifications dans *PatchMatch*. Dans une image rectifiée I comme celle utilisée dans la figure 4.11, nous utilisons une carte avec un code-couleur pour identifier la position de chaque pixel. La couleur rouge indique une zone de faible qualité et la couleur verte indique une zone de forte qualité. Nous affichons la carte de correspondance finale dans le masque Ω (chaque couleur dans le masque indique de quelle zone de I l'information a été prise). Sur l'image de gauche, la carte d'origine contient beaucoup de pixels rouges venant de zones de faible qualité. Cela signifie que l'information copiée dans Ω provient de pixels ayant une faible confiance. Sur l'image de droite, notre carte d'origine utilise plutôt des pixels venant plutôt de zones de forte qualité (les pixels verts sont plus proches de la caméra). Cela signifie, à l'inverse, que l'information copiée dans Ω provient de pixels ayant une forte confiance. Nous avons donc contrôlé la propagation des données de faible qualité dans le deuxième cas, ce qui n'est pas le cas dans le premier cas.

masque Ω. Ce sont donc les données de forte résolutions qui ont été propagées dans le masque dans le deuxième cas, qui correspond à notre approche.

4.5.3 Prise en compte de la résolution dans une approche par analyse statistique

Comme expliqué dans le chapitre 2, l'approche par analyse statistique et plus adaptée que *PatchMatch* pour compléter des textures avec un motif structuré. L' algorithme 6 rappelle les étapes de l'algorithme proposé par HE et SUN [HS12b]. Considérons une image I. L'approche par analyse statistique calcule pour chaque *patch* de la zone connue $O\backslash\Omega$ le *patch* le plus proche selon une fonction de similarité. Le vecteur de translation $\overrightarrow{o}$ liant ces deux *patchs* est appelé *offset* :

$$\overrightarrow{o} = \text{argmin}_{\overrightarrow{u},\|\overrightarrow{u}\|>\tau} \ \text{sim}\left(\Psi_p, \Psi_{p+\overrightarrow{u}})\right) \tag{4.26}$$

où sim est la fonction de similarité et τ est un seuil pour éviter de sélectionner des *patchs* qui sont trop proches du *patch* considéré (notamment la solution évidente, à savoir lui-même). Ensuite, pour chaque *offset*, son occurrence est calculée et les K *offsets* les plus influents sont sélectionnés (ils caractérisent, a priori, la structure du motif de la texture). Enfin, une coupure de graphe est effectuée (6.5) pour créer une carte de label en minimisant l'énergie suivante :

$$E = E_{\text{data}}(\mathbf{p}, \overrightarrow{o}) + E_{\text{reg}}(\mathbf{p}, \mathbf{q}, \overrightarrow{o}_p, \overrightarrow{o}_q), \tag{4.27}$$

où E_{data} mesure le coût de l'association entre l'offset $\overrightarrow{o}$ et le pixel $\mathbf{p}$ et E_{reg} est le terme de régularisation utilisé pour exprimer la consistance visuelle entre les pixels voisins.

Comme *PatchMatch*, cette approche est multi-résolution (les *offsets* sont calculés en basse résolution). Il n'y a là non plus, aucune distinction entre les données de faible et de forte qualité durant la minimisation de l'énergie. De ce fait, des données de faibles qualité peuvent être utilisées pour compléter une zone où une qualité plus forte est attendue. Pour éviter cela, nous introduisons notre critère dans E_{data} de la manière suivante : pour un pixel $\mathbf{p}$ et un *offset* $\overrightarrow{o}$, $E_{\text{data}}(\mathbf{p}, \overrightarrow{o})$ vaut 0 si $\mathbf{p} + \overrightarrow{o}$ n'appartient pas au masque ET si validation$(\mathbf{p}, \mathbf{p} + \overrightarrow{o}, C, \alpha) = 1$ avec $\alpha = 1$. Dans le cas contraire, l'énergie en ce pixel vaut $+\infty$. Les changements sont écrits en rouge dans l'algorithme 6. Concrètement, cela signifie que nous acceptons un pixel candidat uniquement s'il a une valeur connue et s'il a confiance supérieure ou égale à la valeur au pixel à compléter. Notons qu'au moment où le processus d'*inpainting* débute, la confiance est calculée en tout point de l'image rectifiée, y compris les points de Ω.

Dans le cas de textures à forte structure, il peut arriver que nous ne trouvions pas d'*offsets* qui permette de compléter un pixel car seuls les *offsets* qui passent la fonction validation pointent vers un pixel appartenant au masque. Nous n'avons donc pas d'information connue pour compléter le masque. Dans ce cas, nous pouvons autoriser un pixel $\mathbf{p}$ avec une valeur de confiance $\mathcal{C}(\mathbf{p})$ à utiliser un *offset* qui pointe vers un pixel $\mathbf{q}$ avec une confiance $\mathcal{C}(\mathbf{q})$ inférieure mais qui reste proche de $\mathcal{C}(\mathbf{p})$. Pour cela, nous nous servons du paramètre de tolérance α de l'équation 4.25 de la manière suivante. Au début de la coupure de graphe, α est initialisée à 1. Si nous manquons de candidats pour trouver le bon label associé à un pixel masqué, nous diminuons α par une valeur fixée de sorte à réduire la contrainte due à la confiance. On peut ainsi utiliser une donnée d'une qualité légèrement moindre.

Cependant, bien que ces changements dans les deux méthodes d'*inpainting* empêche la génération de flou, elles créent une discontinuité entre les pixels de la zone masquée ayant obtenue une information de qualité élevée et les pixels connus ayant une information de qualité basse. Cela crée une netteté plus forte dans la zone masquée que prévue. Pour retrouver un rendu plus réaliste, nous proposons dans la prochaine sous-section d'altérer le résultat via un flou gaussien variable, dépendant lui aussi de la fonction de confiance.

4.5.4 Flou gaussien variable

Dans cette sous-section, nous proposons une nouvelle application de notre critère de confiance. Nous voulons dégrader l'image obtenue dans la sous-section précédente. En effet, comme nous avons copié des données de qualité haute dans une zone du masque Ω où la confiance attendue est beaucoup moins élevée, une discontinuité se crée entre la qualité de l'information des pixels du masque et ceux qui sont à la périphérie du masque (voir l'image de gauche de la deuxième ligne de la figure 4.13). Si cette discontinuité en soit n'est pas gênante, elle peut créer un rendu trop net du point de vu de l'utilisateur qui verra la différence sur l'image finale entre les données réelles et les données copiées (voir l'image de gauche de la troisième ligne de la figure 4.13).

Pour cela, appliquer un flou basé sur un noyau gaussien est possible. Or, l'appliquer sur toute l'image va dégrader l'information de la même manière, et pour un pixel de haute qualité et pour un pixel de basse qualité.

Pour éviter cela, nous faisons varier l'intensité du flou en fonction de la confiance du pixel considéré : nous appliquons un noyau gaussien varie en fonction de la confiance d'un pixel de l'image définie dans la section 4.4 .
Soit un pixel $\mathbf{p}$ du masque Ω de confiance $\mathcal{C}(\mathbf{p})$. Ce pixel hérite de la valeur d'un pixel $\mathbf{q}$ de confiance $\mathcal{C}(\mathbf{q}) \geqslant \mathcal{C}(\mathbf{p})$ pour éviter toute propagation des données de moindre qualité.

Or, si le pixel $\mathbf{p}$ a une confiance éloignée de $\mathbf{q}$, il reçoit une information de qualité trop élevée par rapport à la qualité attendue en ce pixel. La netteté de la texture au pixel $\mathbf{p}$ sera plus prononcée car supérieure à celle des pixels proches de Ω. Nous effectuons donc notre dégradation par rapport à cette différence de confiance. Si la différence de confiance entre les pixels est faible, la dégradation doit donc être minime. À l'inverse, si cette différence est élevée, la dégradation doit être plus forte. Nous proposons de faire varier l'intensité du flou gaussien en fonction de la différence entre les deux valeurs de confiance.
Rappelons que le flou gaussien est défini comme une convolution entre l'image I et un noyau de convolution G définie de la manière suivante :

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{4.28}$$

où σ est la variance du flou appliqué à l'image. En général, avoir $\sigma < 1$ signifie que l'on veut réduire le bruit. Si $\sigma > 1$, c'est dans le but d'effectuer un flou sur l'image (comme dans notre situation). Plus σ est élevée, plus le flou sera important. Si le flou est défini en un pixel p par un noyau gaussien G_p de variance σ_p, nous définissons σ_p de la façon suivante :

$$\sigma_p = 1 + \beta|\mathcal{C}(c(\mathbf{p})) - \mathcal{C}(\mathbf{p})| \tag{4.29}$$

où $c(p)$ est le pixel de confiance $\mathcal{C}(c(p))$ d'où provient l'information pour remplir p et β est un paramètre qui fait varier l'intensité du flou en fonction de la différence. Actuellement, le choix de β n'est pas automatique. L'automatisation est l'une des perspectives évoquées en fin de chapitre. Nous posons, de manière empirique, $\beta = 2$ Donc le noyau de convolution du flou appliquée en p vaut :

$$G(x,y) = \frac{1}{2\pi\sigma_p^2} e^{-\frac{(x^2+y^2)}{2\sigma_p^2}} \tag{4.30}$$

Nous obtenons aussi le comportement attendu : si $\mathbf{p}$ et $c(\mathbf{p})$ ont une confiance proche, le flou sera faible ; à l'inverse, si les confiances de $\mathbf{p}$ et de $c(\mathbf{p})$ sont éloignées, le flou sera important.

La figure 4.13 montre un exemple d'application de ce flou variable dans une image rectifiée complétée et dans son rendu associé. Nous voyons qu'à la frontière entre la zone connue et masque (deuxième colonne, deuxième et quatrième ligne), nous n'avons plus cette la discontinuité due à la propagation des données de haute qualité dans des zones de basse qualité a été gommée.
Dans la prochaine section, nous montrons des résultats de ces approches modifiées.

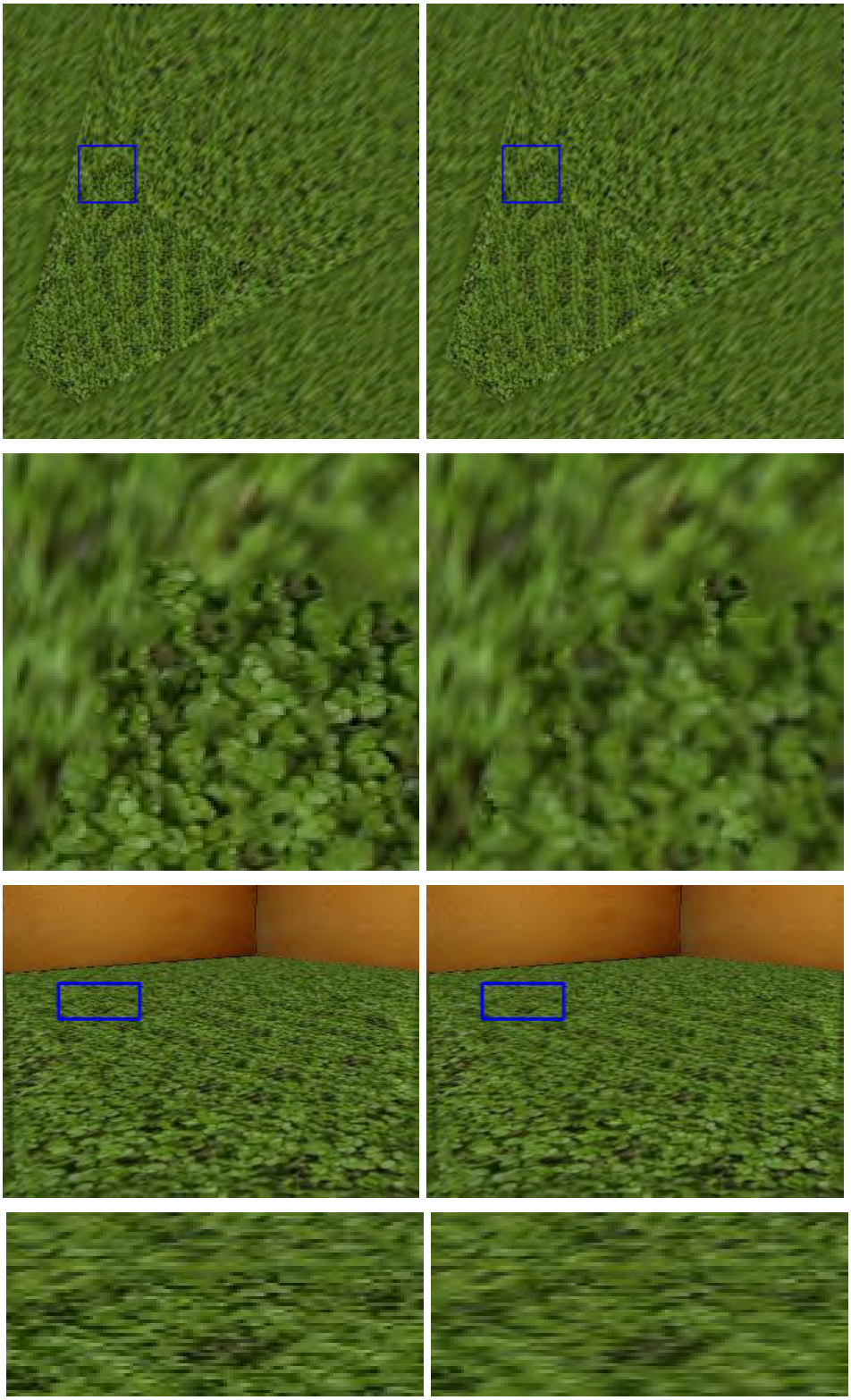


FIGURE 4.13 – Influence du flou gaussien variable sur une image de synthèse. Colonne de gauche : sans flou, colonne de droite : avec flou. Première ligne : image rectifiée complétée avec critère de confiance. Deuxième ligne : zoom sur la zone encadrée en bleu dans l'image au-dessus. Troisième ligne : rendu final avec critère de confiance. Quatrième ligne : zoom sur la zone encadrée en bleu dans l'image au-dessus. Dans la colonne de gauche, nous avons une discontinuité due à la copie des données de haute qualité des zones de masque où l'on attend une qualité moindre. On a donc un rendu trop net dans cette zone. À l'inverse, dans la colonne de droite, la discontinuité a été fortement réduite grâce au flou gaussien variable sans altérer les données de qualité haute. Le rendu final est donc bien plus photo-réaliste.

Algorithm 5 Algorithme *PatchMatch* avec critère de confiance

Entrée: Image I_1, Patch I_2 , Carte de confiance $\mathcal{C}$ ▷ I_1 est l'image de référence, I_2 est l'image patch

Sortie: Carte de correspondances c

Procédure $PatchMatch(I_1, I_2, n_{\text{iter}}, c_{\text{init}}, \omega, \mathcal{C})$
  Si $c_{\text{init}} \neq \varnothing$ **Alors** ▷ Étape d'initialisation
    $c \leftarrow c_{\text{init}}$
  Sinon
    $c \leftarrow$ ALÉA( )
  Fin Si
  Pour $j \leftarrow 1 : n_{\text{iter}}$ **Faire**
    Pour tout $p \in I_1$ **Faire**
      MiseÀJourVoisins($\mathbf{p}$, -1, 0, $\mathcal{C}$) ▷ Étape de propagation
      MiseÀJourVoisins($\mathbf{p}$, 1, 0, $\mathcal{C}$)
      MiseÀJourVoisins($\mathbf{p}$, 0, -1, $\mathcal{C}$)
      MiseÀJourVoisins($\mathbf{p}$, 0, 1, $\mathcal{C}$)
      MiseÀJourAléatoire($\mathbf{p}, \omega$, $\mathcal{C}$) ▷ Étape de randomisation
    Fin Pour
  Fin Pour
Fin Procédure

Fonction MiseÀJourVoisins($\mathbf{p}, dx, dy, \mathcal{C}$)
  $\delta \leftarrow (dx, dy)$
  $q \leftarrow c(\mathbf{p} + \delta)$
  Si $\text{SSD}(\mathbf{p}, \mathbf{q} - \delta) \leqslant \text{SSD}(\mathbf{p}, c(\mathbf{p}))$ **ET** validation$(p, q, \mathcal{C}, 1)$ **Alors**
    $c(\mathbf{p}) \leftarrow \mathbf{q} - \delta$
  Fin Si
Fin Fonction

Fonction MiseÀJourAléatoire($\mathbf{p}, \omega, \mathcal{C}$)
  Pour $j \leftarrow 1 : n_{\text{iter}}$ **Faire**
    $\delta \leftarrow (\text{ALÉA}(1, 2^{-j}\omega), \text{ALÉA}(1, 2^{-j}\omega))$
    $\mathbf{q} \leftarrow c(\mathbf{p}) + \delta$
    Si $\text{SSD}(\mathbf{p}, \mathbf{q}) \leqslant \text{SSD}(\mathbf{p}, c(\mathbf{p}))$ **ET** validation$(\mathbf{p}, \mathbf{q}, \mathcal{C}, 1)$ **Alors**
      $c(\mathbf{p}) \leftarrow \mathbf{q}$
    Fin Si
  Fin Pour
Fin Fonction

Algorithm 6 Algorithme Statistique+Graphcut avec critère de confiance

Entrée: Image I, masque M, ratio r, carte de confiance $\mathcal{C}$
Sortie: Image complétée $I_{\text{inpainted}}$

Procédure StatistiqueGraphcut($I, M, r, \mathcal{C}$)
  $I_r \leftarrow$ réduire(I, r)
  $M_r \leftarrow$ réduire(M, r)
  $O \leftarrow$ calculerOffsets(I_r, M_r) ▷ Étape d'analyse
  $labels \leftarrow$ calculerLabels($I_r, M_r, O, \mathcal{C}$)
  $labels \leftarrow$ aggrandir($labels$)
  $I_{\text{inpainted}} \leftarrow$ appliquerLabels($labels$)
  Retourner $I_{\text{inpainted}}, labels$
Fin Procédure

Fonction calculerLabels($I, M, O, \mathcal{C}$)
  $I_e \leftarrow I_{|\Omega \cup \partial\Omega}$
  Pour tout $\mathbf{p} : (x, y) \in I_e$ **Faire**
    Si $p \in \partial M$ **Alors**
      $labels(\mathbf{p}) \leftarrow 0$ ▷ Associer aux pixels adjacents le vecteur nul
    Sinon
      $labels(\mathbf{p}) \leftarrow$ aléa$(1, |O|)$ ▷ Initialisation aléatoire
    Fin Si
    $labels \leftarrow$ Graphcut($I_e, O, \mathcal{C}, E_d, E_r$)
  Fin Pour
  Retourner labels
Fin Fonction

Fonction $E_d(\mathbf{p}, t, I, M, \mathcal{C})$
  Si $\mathbf{p} \in \partial M$ **OU** $((\mathbf{p} + \mathbf{t}) \notin M)$ **ET** validation$(\mathbf{p}, \mathbf{p} + \mathbf{t}, \mathcal{C}, \alpha)$ **Alors**
    Retourner 0
  Fin Si
  Retourner $+\infty$
Fin Fonction

4.6 Résultats

4.6.1 Implémentation

L'implémentation a été faite en C++. Nous utilisons CImg [Tsc12] pour la gestion des images ainsi que pour l'algorithme *PatchMatch*. Nous utilisons l'implémentation d'interpolation de Thevenaz *et al.* [TBU00] pour la rectification des images. Nous utilisons l'implémentation de Kolmogorov et Zabih [KZ04] de coupure de graphe [BK04 ; BVZ01] pour la labellisation des pixels dans l'approche par *offsets*. Nous partons de l'algorithme d'*inpainting* proposé au chapitre 3. Les étapes gérées par *PatchMatch* et les *offsets* ont été modifiées pour tenir du critère de confiance de la section 4.5.

4.6.2 Résultats sur des images rectifiées

La figure 4.14 montre les résultats de deux images rectifiées contenant des textures de synthèse : une texture irrégulière et une texture régulière. Le masque, en rouge, a été défini en supprimant des objets (voir la première ligne de la figure 4.14). Nous utilisons l'approche classique d'*inpainting* par l'algorithme de *PatchMatch* pour le premier cas (première colonne, deuxième ligne) et l'approche par *offsets* dans le second cas (deuxième colonne, deuxième ligne), et par extension, l'approche de l'approche du chapitre 3. La seconde ligne montre que le rendu sans critère est très flou car la donnée de faible qualité a été propagée dans une zone du masque on l'on attend une qualité plus élevée. La troisième ligne utilise l'algorithme proposé dans la section 4.5 basé sur notre critère de confiance et la fonction de validation ; on constate que la complétion est effectuée sans propager des données de basse résolution. Cependant, dans le cas de la texture de gauche, on constate une répétition d'un même motif. Cela est dû au fait que peu d'information de forte qualité est disponible. La probabilité de copier le même motif est donc élevé.

4.6.3 Résultats sur des rendus finaux

La figure 4.15 montre les résultats concernant le rendu final de notre processus de Réalité Diminuée sur une image de synthèse (première ligne, image de gauche) et de deux image prises en situation réelle (première ligne, images du milieu et de droite). La méthode d'*inpainting* est choisi avec la méthode de classification vue au chapitre 3. Pour les images prises en situation réelle, la luminosité n'est pas contrôlée. Nous avons donc appliqué la méthode qui sera expliquée dans le chapitre 5 pour normaliser et adapter la luminosité présente dans l'image rectifiée. Cela permet d'avoir un rendu plus réaliste. Pour chaque cas (première colonne), nous apercevons que la donnée de basse qualité a été propagée dans la zone où une qualité élevée est attendue. À l'inverse, le fait d'ajouter notre critère via la fonction validation dans les deux approches d'*inpainting* permet de mieux gérer la propagation.

Notre implémentation donne de bon résultats avec les textures irrégulières, même dans des situations complexes. Dans le cas de la texture régulière, nous ajustons le paramètre α comme expliqué dans la section 4.5. Nous appliquons également le flou variable pour contrôler la propagation des données de haute qualité dans les zones de basse qualité.

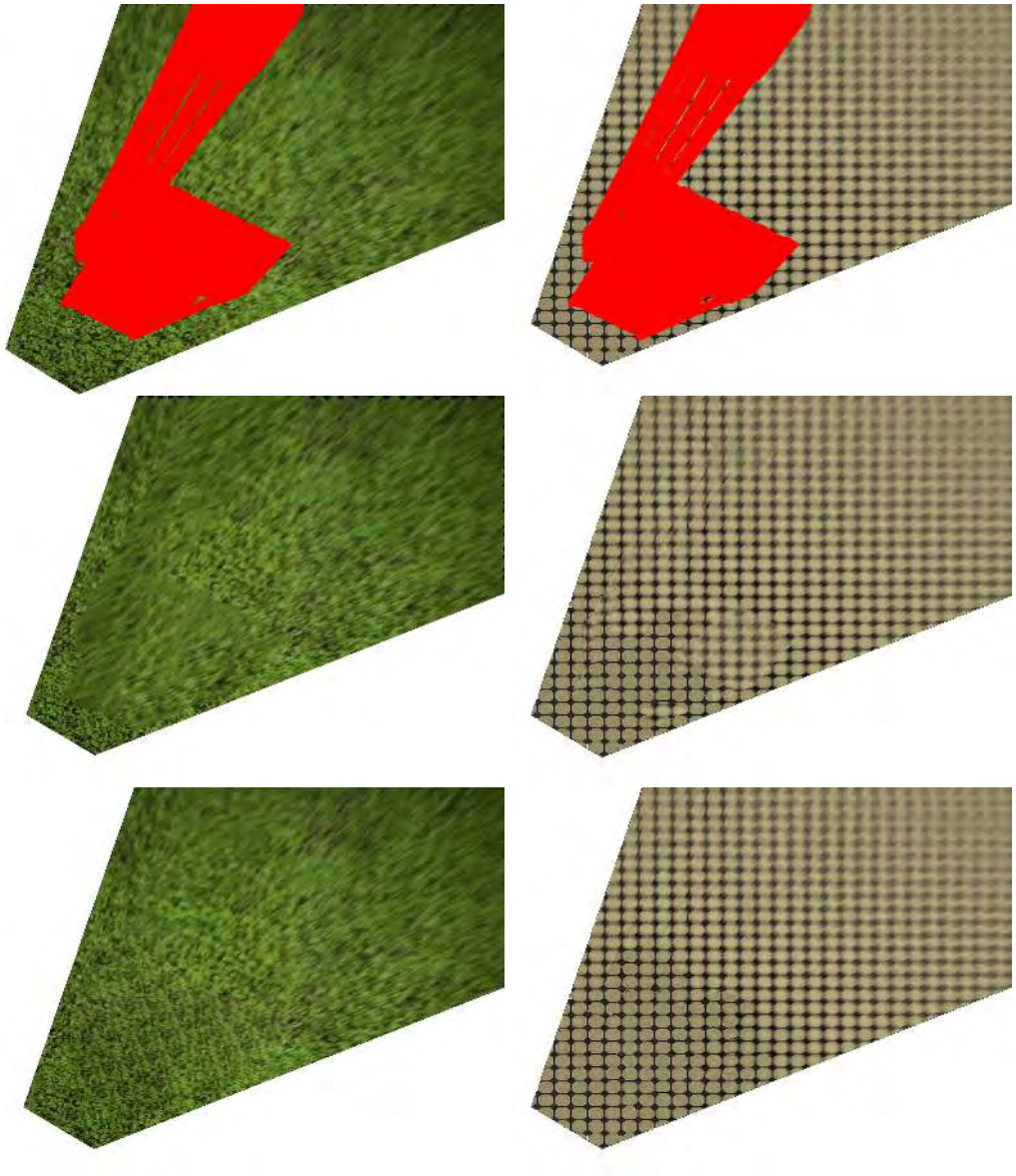


FIGURE 4.14 – Résultats de l'approche basée *PatchMatch* (gauche) et de l'approche basée *offsets* (droite) sur une zone vide coloriée en rouge (première ligne) sans le critère validation (deuxième ligne) et avec un critère de validation (troisième ligne).

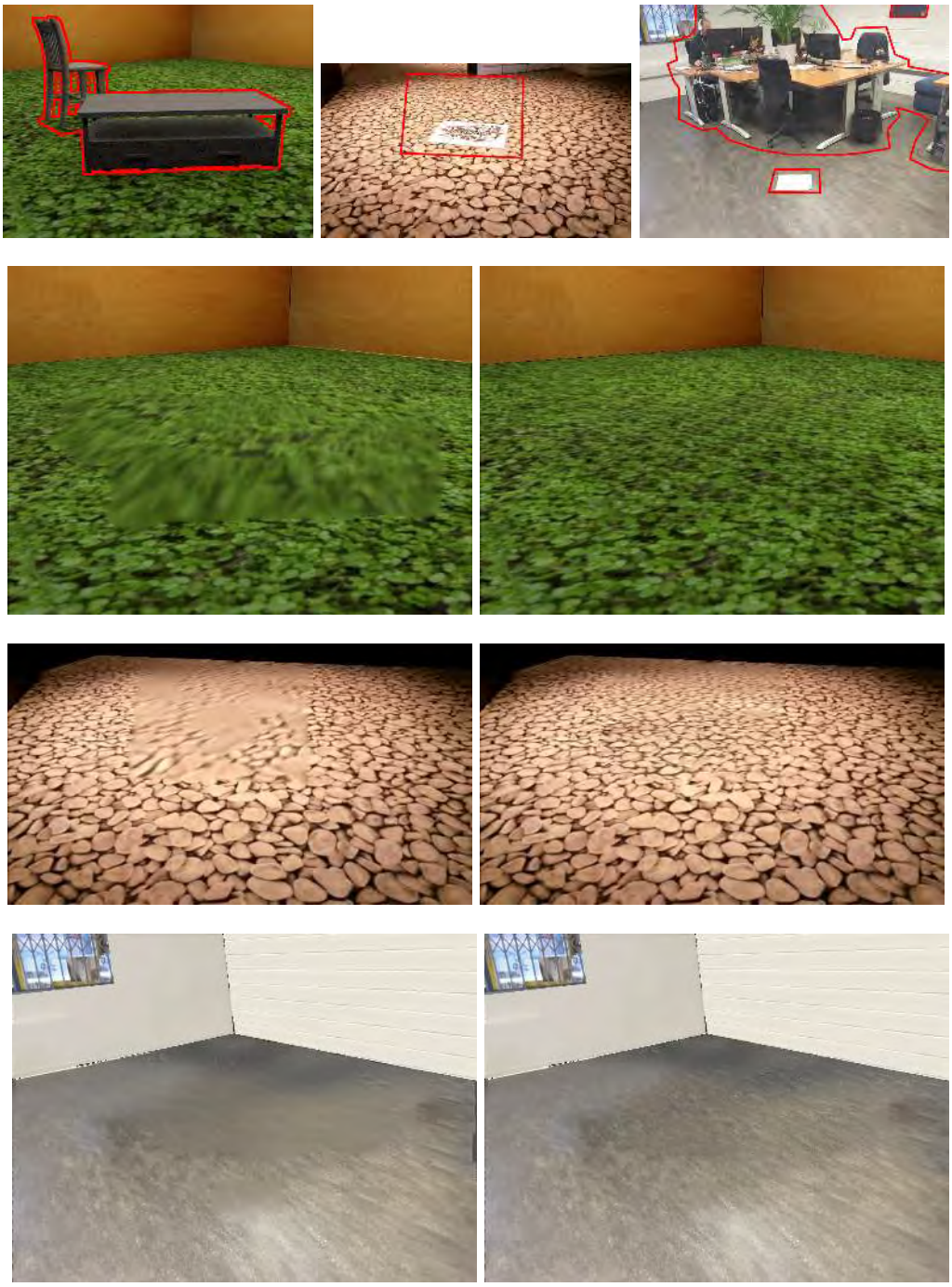


FIGURE 4.15 – Résultats du processus global de Réalité Diminuée. Première ligne : images d'entrée avec la zone à effacer encadrée en rouge. Pour les lignes suivantes. Première colonne : résultat avec l'approche de l'état de l'art. Deuxième colonne : résultat avec notre approche. Pour le troisième cas de figure, une approche d'*inpainting* couplée au critère de confiance est appliquée au plan du mur et du mur de gauche tandis qu'une synthèse par coupure de graphe [Kwa+03] est appliquée au mur de droite.

4.7 Conclusion

Dans ce chapitre, nous avons proposé un critère de confiance pour mieux guider la sélection d'un *patch* dans des méthodes d'*inpainting* dans les images rectifiées, générées grâce à la géométrie planaire par morceaux de la scène. Cela permet ainsi de gérer la propagation des données de basse qualité.
Nous avons montré également que ces traitements supplémentaires dans l'image rectifiée évite la génération de flou dans le rendu final. Nous avons en outre utilisé le critère de confiance a posteriori pour retrouver une continuité de la qualité dans la zone complétée.
Une perspective possible concerne la partie sur le flou gaussien variable énoncé dans la sous-section 4.5.4. Le paramètre d'intensité β défini dans l'équation 4.29 est actuellement calculé empiriquement. La prochaine étape sera de trouver un moyen qui permette de le calculer automatiquement.
Une autre perspective est de contrôler la répétition d'un même motif quand nous avons peu d'information de forte qualité. Pour cela, il est possible de pénaliser le choix d'un motif à partir du nombre d'occurrence d'une correspondance dans *PatchMatch*.

Nous avons traité de la problématique de la complétion en fonction du type de la texture dans le chapitre 3 puis de celle concernant la différence de résolution dans ce chapitre. Dans le chapitre suivant, nous abordons la problématique de la luminosité et des solutions proposées pour gérer ses variations dans le processus de diminution.

Publication associée

Julien Fayer *et al.*. « Radiometric confidence criterion for patch-based inpainting ». In : *Proceedings of the 24th International Conference on Pattern Recognition (ICPR 2018)*. Août 2018, p. 2723–2728. DOI : 10.1109/ICPR.2018.8545350

Chapitre 5

Luminosité : modélisation et traitement dans la complétion

Sommaire

5.1 Introduction

Après avoir abordé les contraintes liées à la structure de la texture et à la différence de résolution, nous traitons dans ce chapitre de la nécessité de gérer la luminosité dans la complétion des images rectifiées. Les environnements d'intérieur possèdent différentes sources de lumières qu'elles soient naturelles (lumière extérieure) ou artificielles (lampe). Celles-ci modifient la perception que nous avons de l'environnement et sont particulièrement importantes pour le réalisme. De plus, elles impactent l'exposition de la caméra selon son point de vue.

Les variations de l'éclairage dans une image peuvent être représentées par le modèle de Phong [Pho75]. Celui-ci calcule la luminosité en un point de l'image rendue à partir des paramètres suivants : la position du point observé par la caméra, la position des

sources lumineuses, la couleur et l'intensité des sources lumineuses ainsi que les propriétés physiques de l'objet (lisse, rugueux, soyeux, *etc.*).
Ce modèle est constitué de 3 termes d'illumination :

- L'illumination ambiante modélise la lumière réfléchie par les autres objets dans la scène ; c'est la lumière de l'environnement.
- La composante diffuse caractérise la réflexion des rayons qui frappent un objet de la scène. Elle tient compte de l'inclinaison avec laquelle la lumière incidente arrive sur la surface car la quantité de lumière en un point de la surface dépend de l'angle entre vecteur normal et du vecteur incident de la lumière. Elle suppose aussi que l'intensité est la même quelque soit la direction que prend le rayon réfléchi ; elle ne dépend donc pas de la position de l'observateur.
- La composante spéculaire tient compte du fait que, pour certains matériaux, l'illumination change en fonction de la position de l'observateur, pour des surfaces « brillantes », comme le plastique. Cela crée des tâches brillantes, appelées tâches spéculaires, dans les cas où ces matériaux sont éclairés par une lumière forte.

La somme de ces 3 termes forment le modèle de Phong [Pho75].
Dans ce chapitre, nous expliquons dans la section 5.2 comment nous gérons l'illumination ambiante en adaptant l'exposition en utilisant des méthodes de l'état de l'art. Ensuite, dans les sections 5.3 et 5.4, pour gérer la composante diffuse, nous proposons une méthode originale pour gérer la variation de luminosité, notamment en terme de propagation au sein d'une image rectifiée. Enfin, dans la section 5.5, nous nous servons de deux méthodes de la littérature [STB18 ; MT14] pour gérer la composante spéculaire.

5.2 Calibrage de l'exposition

Dans cette section, nous traitons du problème de calibrage de l'exposition dans notre problématique de Réalité Diminuée. Nous expliquons comment ce problème est traitée à partir des méthodes existantes.

5.2.1 Problématique

Comme expliqué dans le chapitre 4, chaque image rectifiée représentant un plan de la pièce d'intérieur est la concaténation de plusieurs images. Ces images proviennent d'une même caméra ayant une position et une orientation différentes. Cependant, cette caméra subit une exposition différente selon son orientation par rapport aux différentes sources de lumière. Une même texture va paraître claire ou sombre selon le degré d'exposition de la caméra. Si cette différente d'exposition entre plusieurs images n'est pas gérée durant la complétion, des artefacts peuvent apparaître. Un artefact est, par exemple, la frontière entre deux zones complétées avec des données venant d'images d'expositions différentes comme montré dans la figure 5.1.

5.2.2 Modèle complet

Nous rappelons le modèle complet d'exposition proposé par Goldman [Gol10]. Nous notons I une image issue d'une caméra C. Cette caméra est soumise à une exposition $\alpha \in \mathbb{R}$ qui représente la quantité de lumière atteignant un capteur numérique, par unité de surface. Elle possède également une courbe de réponse nommée $R : \mathbb{R} \to [0, 255]$. Cette fonction définit la couleur du pixel dans l'image en fonction de la quantité de lumière arrivant sur le capteur. En raison de la loi de Bouguer, le flux lumineux reçu varie selon l'incidence du rayon lumineux par rapport au plan de la caméra. Cet effet

FIGURE 5.1 – Différence d'exposition. Première image : image rectifiée construite à partir de deux prises de vue avec des expositions différentes. Deuxième image : masque Ω. Troisième image : résultat de l'*inpainting* à partir de ces deux images de deux expositions différentes. On y voit que la complétion utilise les deux zones de différentes expositions. Sans gestion, cela conduit à une frontière au sein de la zone masquée ce qui n'est vraiment pas souhaitable sur une région à l'origine homogène.

s'appelle le vignettage. Le vignettage est modélisé par la fonction de vignettage M : $\mathbb{R} \to \mathbb{R}, r \mapsto M(r)$ où r est le rayon entre le pixel p et le centre de l'image. À chaque pixel p est associée une valeur $L(p)$ nommée radiance indépendante de l'exposition ou du vignettage. Le modèle complet est le suivant :

$$I(p) = R(\alpha\, L(p)\, M(r_p)), \tag{5.1}$$

où $I(p)$ est la valeur du pixel p dans l'image I.

Par la suite, comme les micro-lentilles actuelles réduisent l'effet de vignettage sur les photos, nous considérons le terme de vignettage M négligeable et la courbe de réponse R connue dans l'équation 5.1 , nous avons donc l'équation :

$$I(p) = R(\alpha\, L(p)). \tag{5.2}$$

Si la courbe de réponse n'est pas connue, GOLDMAN [Gol10] montre que cette formulation du problème définit des familles de paramètres (exposition, vignettage, radiance, courbe de réponse) qui produisent les même valeurs de pixels dans l'image. Il suffit alors de trouver n'importe quelle solution de cette famille pour retrouver les vraies valeurs des paramètres. Il reste finalement $L(p)$ et α qui sont encore inconnus. Dans la prochaine section, nous expliquons comment déterminer ces valeurs.

5.2.3 Calcul de l'exposition

Nous considérons un ensemble de prises de vues I_i dont nous cherchons une exposition α_i. Nous notons $I_{\Delta,i}$ la zone de l'image I_i transformée par l'homographie de rectification $\mathtt{H}_i$ associée au plan Δ. Nous rappelons que l'image rectifiée globale I_Δ est la concaténation de toutes les images rectifiées $I_{\Delta,i}$. Nous associons pour chaque pixel $p \in I_\Delta$ la valeur radiance $L(p)$ définie dans la sous-section précédente. Cette valeur correspond à la couleur du pixel indépendamment de l'exposition. L'avantage de considérer I_Δ est de disposer de plusieurs valeurs pour un pixel de I_Δ construites à partir de la même valeur radiance mais de plusieurs coefficients d'exposition.

La recherche de ces expositions ressemble à la méthode utilisée dans le recollage d'images panoramiques [Gol10]. Une correspondance y est cherchée entre les pixels des différents points de vue. Les pixels (qui représentent le même point d'intérêt) peuvent

donc avoir différentes valeurs alors qu'ils ont la même radiance dans l'image panoramique. Tous les pixels représentant le même point d'intérêt dans différentes images sont projetés sur le même pixel de l'image panoramique. La radiance de chaque point d'intérêt et l'exposition de chaque point de vue peuvent donc être retrouvées simultanément en minimisant l'erreur de projection dans l'image panoramique. L'erreur à minimiser, pour tous les pixels p et pour toutes les prises de vue $\{i\}$, est la suivante :

$$\min_{\alpha_i, L(p)} \sum_{i,p} \mathrm{d}\left(X_p^i, R(\alpha_i L(p))\right), \tag{5.3}$$

où d est une fonction de distance, ici $|\cdot|^2$, X_p^i est la valeur du pixel p dans le point de vue i, α_i est le coefficient d'exposition de la caméra associé au point de vue i et $L(p)$ est la radiance d'un pixel p. La résolution de ce problème aux moindres carrés se fait en utilisant le complément de Schur [Tri+00].

Dans notre cas, le modèle de correspondance est basé sur des considérations géométriques [ZCC16]. Deux pixels p et q de deux images I_i et I_j vont être en correspondance s'ils sont projetés au même endroit dans I_Δ. Ainsi, pour pouvoir minimiser cette erreur, il est nécessaire d'avoir des zones où au moins deux images sont projetées dedans.

La formulation du problème ressemble à celle de l'ajustement de faisceaux (ou *bundle adjustement* [HZ04]) :

$$\min_{\mathtt{T}_i, u_p} \sum_{i,p} \mathrm{d}\left(X_p^i, \mathtt{K}\,\mathtt{T}_i.\mathbf{u}_p\right), \tag{5.4}$$

où $\mathtt{K}$ est la matrice des paramètres intrinsèques, $\mathtt{T}_i$ est la matrice des paramètres extrinsèques de la caméra à la position i et $\mathbf{u}_p$ la position du point 3D dans le référentiel monde et X_p^i est la position du point 3D dans le plan-image de la caméra à la position i.
Nous adaptons donc l'équation 5.3 à notre problème :

$$\min_{\alpha_i, L(p)} \sum_{i,p} \mathrm{d}\left(I_{\Delta,i}(p), R(\alpha_i L(p))\right), \tag{5.5}$$

où $I_{\Delta,i}(p)$ est la valeur du pixel p dans l'image rectifiée $I_{\Delta,i}$ associée au point de vue i, avec α_i comme coefficient d'exposition, $L(p)$ est la radiance du pixel p dans l'image rectifiée I_Δ.

La figure 5.2 montre un exemple du calcul de l'exposition sur une image rectifiée composée de deux images d'expositions différentes. Le résultat montre bien que l'exposition a correctement été estimée. Cela permet d'avoir une bonne base pour la suite du processus de gestion de la luminosité.

Une fois l'exposition trouvée pour chaque image $I_{\Delta,i}$, ses valeurs sont divisées par son exposition :

$$I_{\Delta,i} \leftarrow \frac{I_{\Delta,i}}{\alpha_i}. \tag{5.6}$$

Le terme ambiant est traité. Cependant, comme expliqué dans le chapitre 1, il reste à corriger la variation de luminosité, à travers les composantes diffuse et spéculaire, si on veut pouvoir appliquer un processus d'*inpainting* dans les meilleures conditions. Nous traitons de cela dans la prochaine section.

5.3 Propagation de la luminosité : méthodes actuelles

Nous abordons dans cette section les méthodes qui proposent une gestion de la luminosité au sein d'un scénario de Réalité Diminuée.

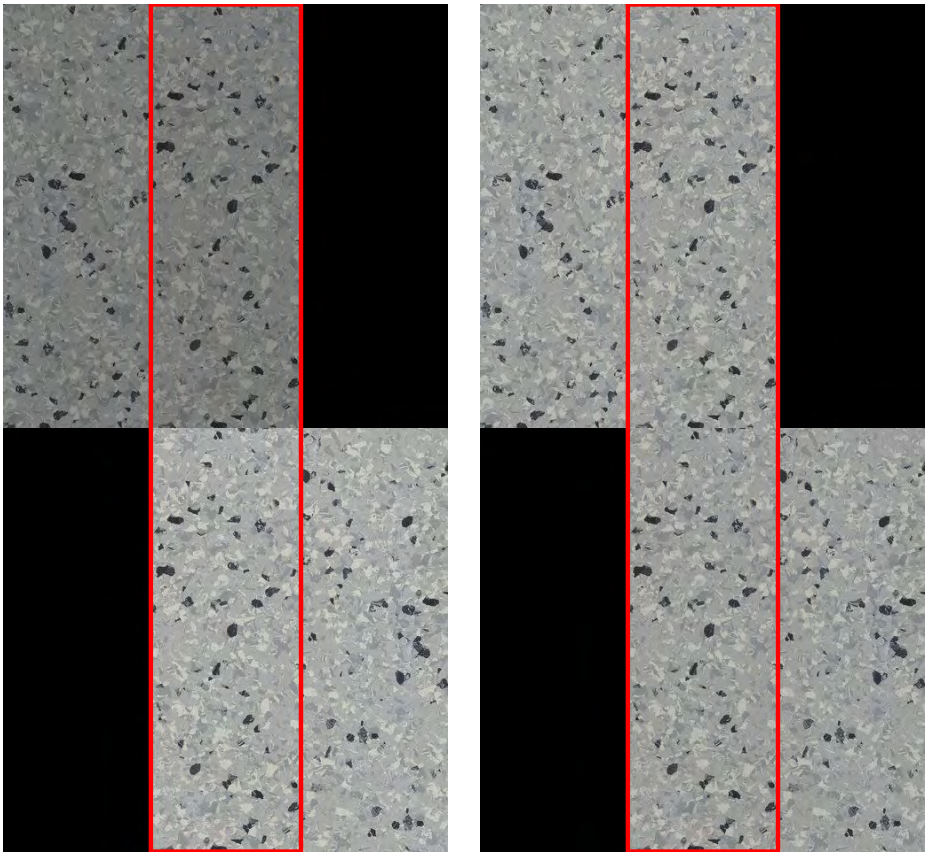


FIGURE 5.2 – Exemple de gestion de l'exposition dans le cas de deux images rectifiées (une "foncée" sur la ligne du haut, une "claire" sur la ligne du bas) en utilisant la zone centrale (encadrée en rouge) commune aux deux images comme données pour la minimisation de l'erreur. La colonne de gauche montre les deux images rectifiées sans l'adaptation de l'exposition. La colonne de droite montre les deux images rectifiées avec l'adaptation de l'exposition. Nous montrons la gestion dans le cas de plusieurs images dans le chapitre 7.

À ce stade, nous disposons pour un plan Δ de son image rectifiée dans laquelle les variations d'exposition globales ont été effacées :

$$I_\Delta = \bigcup_i \frac{I_{\Delta,i}}{\alpha_i}. \tag{5.7}$$

Cependant, il faut maintenant gérer les différences de luminosité plus locales présentes au sein de cette image. Sans gestion, la complétion ne sera pas optimale comme montrée dans la première image de la figure 5.3. En effet, les techniques traditionnelles d'*inpainting* ne vont pas prendre en compte :

- les différences entre les pixels ayant une forte luminosité et ceux ayant une faible luminosité,
- l'absence probable de pixels ayant une luminosité intermédiaire pour compléter la zone masquée.

Il existe trois méthodes pour gérer cette variation de luminosité dans la complétion. KAWAI *et al.* [KSY15] prennent en compte directement la gestion de la luminosité dans leur approche d'*inpainting*. Elle consiste à minimiser l'énergie suivante dans une zone Ω'

encadrant le masque Ω :

$$E = \sum_{x_i \in \Omega'} \omega_{x_i} \left(\text{sim}_1(x_i, x_j, T_{x_i,x_j}) + \lambda \, \text{sim}_2(x_i, x_j) \right), \tag{5.8}$$

où x_i est un pixel de Ω', x_j est un pixel de $O \backslash \Omega'$, ω_{x_i} et λ sont des poids, T_{x_i,x_j} est une matrice de transformation entre deux pixels x_i et x_j. La prise en compte de la luminosité se trouve dans la définition de la fonction de similarité sim_1 :

$$\text{sim}_1(x_i, x_j, T_{x_i,x_j}) = \sum_{p \in W} \left(I(p) - \alpha_{x_i,x_j} I(T_{x_i,x_j} p) \right)^2, \tag{5.9}$$

où W est l'ensemble des pixels du *patch* centré en x_i et α_{x_i,x_j} est le coefficient de luminosité. Ce dernier ajuste la luminosité de la texture dans la zone de données pour éviter les différences excessives entre celle-ci et la zone masquée. La fonction de similarité sim_2 est définie de la manière suivante :

$$\text{sim}_2(x_i, x_j) = \frac{\| W \|}{1 + e^{-K \| x_i - x_j \| - X_0}}, \tag{5.10}$$

où $\| W \|$ est le nombre de pixels dans un *patch* ; K et X_0 sont des hyper-paramètres globaux (fixés de manière empirique pour toutes les images). Cette fonction favorise les pixels qui sont proches du masque Ω. Le coefficient α_{x_i,x_j} est défini, en pratique, comme le ratio entre les moyennes des valeurs des pixels situés autour de x_i et de x_j. L'avantage de cette méthode est d'intégrer directement la prise en compte de la luminosité dans la fonction de similarité pour compléter le masque. L'inconvénient est d'avoir, cependant, des hyper-paramètres dont il est nécessaire d'ajuster si on veut avoir des résultats convenables.

Siltanen [Sil15] dissocie la gestion de la luminosité de la complétion de la texture. L'image I_Δ est d'abord normalisée par l'opération suivante :

$$I_{\Delta,n} = I_\Delta + I_{\Delta,m} - A, \tag{5.11}$$

où $I_{\Delta,m}$ est l'image qui à chaque pixel p associe sa valeur médiane d'un patch centré en p, $A \in \mathbb{R}$ est la moyenne des valeurs de l'image $I_{\Delta,m}$. Nous avons donc $I_{\Delta,n}$ qui ne contient que la texture. La carte de luminosité est obtenu en soustrayant $I_{\Delta,n}$ à I_Δ. Le procédé de complétion est appliqué ensuite à $I_{\Delta,n}$. Pour retrouver ensuite les valeurs de la carte de luminosité dans Ω, une adaptation de luminosité est effectuée. Celle-ci va utiliser les valeurs d'un ensemble de points de contrôle situés autour de Ω pour la propager dans Ω. Nous nous baserons pour la suite de ce chapitre sur la méthode de Siltanen [Sil15].L'avantage de cette méthode est de faire abstraction des variations de luminosité dans l'*inpainting*. En effet, l'image qui est traitée par l'*inpainting* est l'image normalisée $I_{\Delta,n}$. Or, celle-ci, ainsi construite, ne possède pas de variation de luminosité. Il n'est donc pas nécessaire de modifier l'approche d'*inpainting* du chapitre 3. La propagation de la luminosité est effectué dans un traitement indépendant. Ce traitement de la propagation de la luminosité est détaillée dans la sous-section suivante. L'autre avantage est qu'il est simple d'intégrer cette méthode dans notre processus de Réalité Diminuée expliqué dans le chapitre 6.

Enfin, Herling et Broll [HB14] proposent de sélectionner un sous ensemble de pixels c de la frontière extérieure $\partial\Omega_e$ qui permettent de représenter/modéliser les variations de la luminosité sur la frontière du masque. À partir des valeurs connues d'une trame clef où les variations de luminosité sont connues en ces pixels c, une fonction de

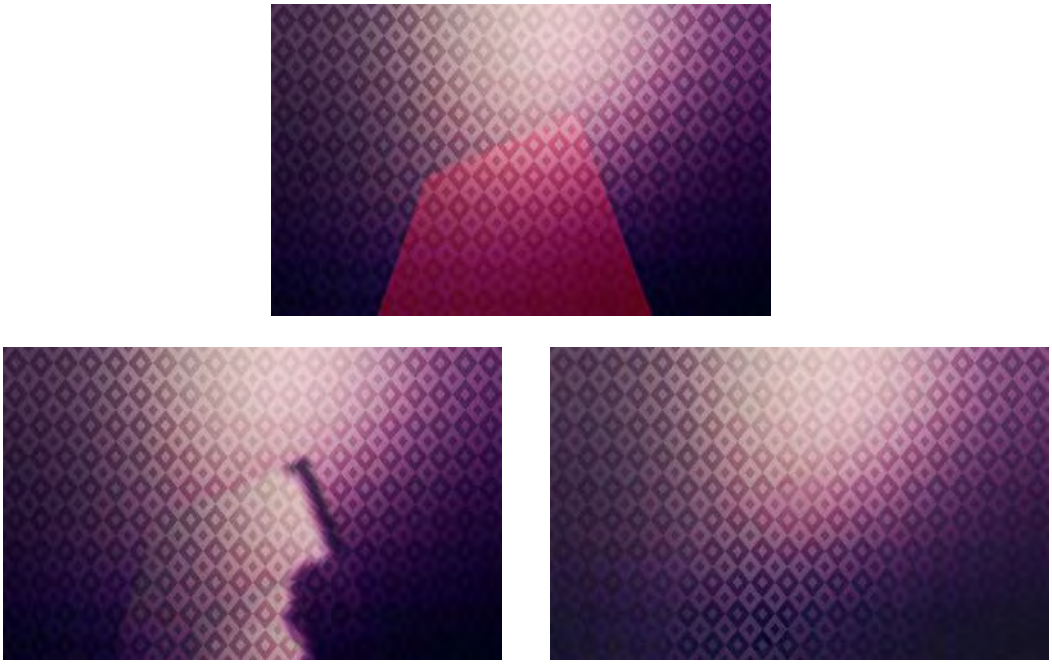


FIGURE 5.3 – Gestion de la luminosité durant la complétion. La ligne du haut montre la vérité terrain avec le masque Ω colorié en rouge. La ligne du bas montre les résultat d'*inpainting* : sans gestion de luminosité (image de gauche) et avec gestion de la luminosité (image de droite). On constate que sans gestion de la luminosité, la structure de la texture est respectée mais des artefacts d'intensité lumineuse créés dans le masque rendent le résultat exécrable. À l'inverse, le résultat est bien meilleur et beaucoup plus réaliste si la gestion de la luminosité est prise en compte dans le processus d'*inpainting*.

correction est estimée entre ces mêmes pixels dans la trame courante :

$$\psi(c) = \frac{1}{\theta(c)} \sum_{j=1}^{|\Omega_e^K|} \left(I_K(b_j^K) - I_F(b_j^F) \right) e^{-|c - b_j^K|^{\frac{1}{2}}}, \tag{5.12}$$

avec Ω_e^K représentant la frontière extérieure dans la trame clef et contenant les points $b_1^K \dots b_{|\Omega_e^K|}^K$; Ω_e^F représentant la frontière dans la trame actuelle et contenant les points $b_1^F \dots b_{|\Omega_e^F|}^F$. Le terme $\theta(c)$ est un facteur de normalisation défini de la manière suivante :

$$\theta(c) = \sum_{j=1}^{\Omega_e^K} e^{-|c + b_j^K|^{\frac{1}{2}}}. \tag{5.13}$$

Chaque pixel p de la zone masquée est ensuite corrigé par une interpolation bilinéaire en considérant les coefficients des points du contour les plus proches suivant les quatre directions cartographiques.
Il n'est donc pas nécessaire de modifier notre approche du chapitre 3 pour tenir compte de la luminosité vu qu'il suffit d'appliquer la normalisation dans l'image rectifiée I_Δ. Cependant, la propagation de luminosité dans le masque Ω de SILTANEN [Sil15] a plusieurs limitations et peut être améliorée. Nous abordons ces limites et notre amélioration dans la section suivante.

5.4 Complétion de la carte de luminosité

Dans cette section, nous abordons la manière de compléter la carte de luminosité générée à partir de la méthode de SILTANEN [Sil15] énoncée dans la sous-section précédente. Nous définissons et mettons en évidence les propriétés de ces cartes de luminosité. Nous décrivons ensuite les méthodes actuellement utilisées pour compléter la zone masquée Ω de la carte de luminosité. Enfin, nous présentons notre méthode pour compléter les pixels du masque Ω dans la carte de luminosité.

5.4.1 Définition et propriétés

Soit U un domaine de $\mathbb{N}^2$, nous définissons $\mathrm{C}_{\mathrm{lum}}$ comme l'image à valeurs réelles suivante :

$$\mathrm{C}_{\mathrm{lum}} : U \to \mathbb{R} \tag{5.14}$$

$$(x, y) \mapsto \mathrm{C}_{\mathrm{lum}}(x, y) \tag{5.15}$$

$\mathrm{C}_{\mathrm{lum}}$ a les propriétés suivantes :

1. $\mathrm{C}_{\mathrm{lum}}$ est bornée sur U, on notera $z_{\min}$ et $z_{\max}$ son minimum et son maximum, tous les deux atteints en un ou plusieurs points ,
2. $\mathrm{C}_{\mathrm{lum}}$ est de classe $\mathcal{C}^1$.

Nous considérons les lignes de niveaux (ou isocontours) définies de la manière suivante :

$$H_z = \left\{(x, y) \in \mathbb{R}^2, \mathrm{C}_{\mathrm{lum}}(x, y) - z = 0\right\} \tag{5.16}$$

pour un ensemble de valeurs échantillonnées dans l'intervalle $z \in [z_{min}, z_{max}]$.

5.4.2 Construction de la carte de luminosité

Pour commencer, l'image I_Δ est définie de la manière suivante :

$$I_\Delta = T_\Delta + \mathrm{C}_{\mathrm{lum}\Delta}, \tag{5.17}$$

où T_Δ est la carte de texture (ou image normalisée) et $\mathrm{C}_{\mathrm{lum}\Delta}$ est l'image de variation de luminosité (ou carte de luminosité). La séparation est obtenue à partir de l'équation 5.11. La carte de luminosité $\mathrm{C}_{\mathrm{lum}\Delta}$ est obtenue partir de l'image normalisée $I_{\Delta,n}$, qui est soustraite à I_Δ :

$$\mathrm{C}_{\mathrm{lum}\Delta} = I_\Delta - I_{\Delta,n} = A - I_{\Delta,m}, \tag{5.18}$$

où, pour rappel, $I_{\Delta,m}$ est l'image qui contient, pour chaque pixel, la valeur médiane d'un patch centré en celui-ci et $A \in \mathbb{R}$ est la moyenne des valeurs de l'image $I_{\Delta,m}$. Cette séparation permet de traiter séparément la complétion de la texture de la complétion de la luminosité. En effet, les deux cartes, par leur propriétés différentes, demandent chacune une méthode de complétion différente :

— La complétion de T_Δ est effectuée par la méthode d'*inpainting* présentée au chapitre 3.

— La complétion de $\mathrm{C}_{\mathrm{lum}\Delta}$ est effectuée par une méthode différente. Celle-ci, au lieu de faire des copies de *patchs* dans le masque Ω, va interpoler les valeurs du masque à partir des informations d'intensité des pixels situés sur la frontière extérieure $\partial\Omega_e$.

Une fois les pixels de Ω complétés dans T_Δ et $\mathrm{C}_{\mathrm{lum}\Delta}$, nous retrouvons I_Δ par simple addition. La figure 5.4 montre un exemple des types de cartes, obtenue à partir d'une image rectifiée.

Nous présentons maintenant les méthodes couramment utilisées pour la zone masquée Ω de $\mathrm{C}_{\mathrm{lum}\Delta}$.

FIGURE 5.4 – Exemple de carte de texture (image de gauche) et de carte de luminosité (image du milieu), extraites à partir d'une image rectifiée (image de droite) via l'équation 5.11.

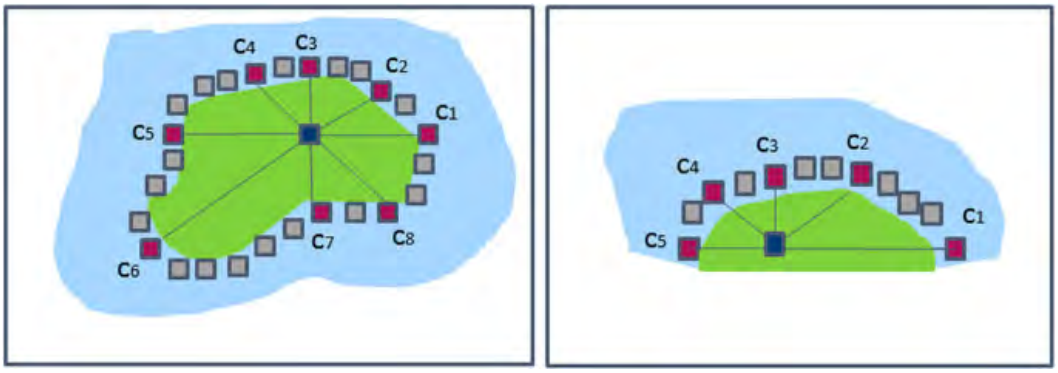


FIGURE 5.5 – Sélection des points de contrôle selon les 8 directions (figure extraite de SILTANEN [Sil15]). Dans l'image de gauche, le masque Ω est centré dans la zone connue (coloriée en bleu), la valeur de chaque pixel du masque est calculée à partir des valeurs de 8 points de contrôle. À l'inverse, dans l'image de droite, le masque est au bord de la zone connue, les pixels n'ont donc que 4 points de contrôle ce qui peut poser problème dans la propagation des valeurs de ces derniers dans le masque.

5.4.3 Méthodes existantes

Nous présentons les deux principales méthodes utilisées pour corriger la variation de luminosité dans un cadre de scénario de Réalité Diminuée.
SILTANEN [Sil15] propose d'utiliser un ensemble de points de contrôle c_i situés autour du masque Ω pour propager leur valeur dans le masque. Ces points sont sélectionnés de façon à être répartis de manière uniforme autour du masque. Ensuite, la luminosité de chaque point du masque $p \in \Omega$ est calculée comme une combinaison affine de coefficients inversement proportionnel [She68] à la distance entre p et chacun des points de contrôle :

$$I(p) = \frac{\sum_{i=1}^{n} \| p - c_i \|^{-4} I(c_i)}{\sum_{i=1}^{n} \| p - c_i \|^{-4}}, \tag{5.19}$$

où n est le nombre de points de contrôle considérés et $\| \cdot \|$ est la norme euclidienne. Le choix d'utiliser la puissance -4 est justifiée de manière empirique dans la mesure où celle-ci apporte une atténuation naturelle des sources de lumière dans des scènes d'intérieur. En plus, pour réduire le temps de calcul, ils proposent de réduire le nombre de points de contrôle utilisés par pixel. Pour cela, ils ne considèrent que , pour un pixel p, les huit points de contrôle les plus proche de p dans les 8 directions majeures comme montré dans la figure 5.5.

La figure 5.6 montre un exemple de résultat par cette méthode. On constate en effet que le rendu après adaptation (image du bas) est bien plus réaliste que le résultat sans adaptation (image du haut).

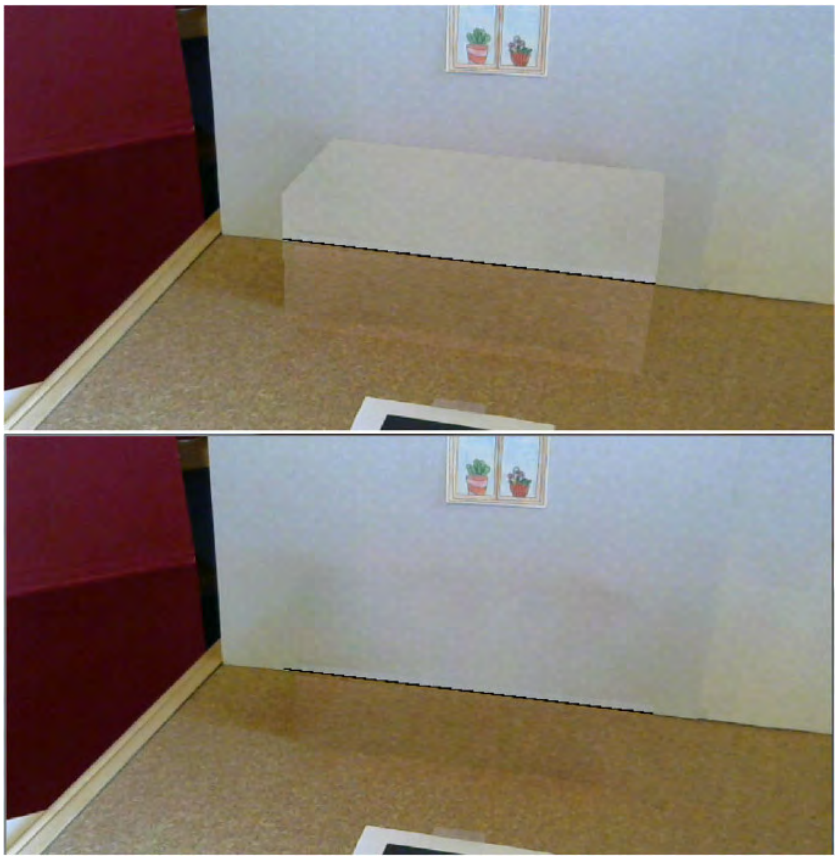


FIGURE 5.6 – Adaptation de la luminosité sur une image complétée (figure extraite de SILTANEN [Sil15]). Sur l'image du haut, on constate clairement la frontière entre le masque Ω et la zone connue car la texture normalisée n'a pas été traitée pour tenir compte de la luminosité. À l'inverse, l'image du bas, pour laquelle la luminosité a été calculée explicitement, fournit un rendu bien plus réaliste.

Si les résultats sont convenables, on constate que la luminosité n'est pas propagée de manière rigoureuse, surtout pour des masques de grande taille. De fait, l'approche de SILTANEN [Sil15] peut être analysée comme une approche d'*inpainting* par diffusion simplifiée : si l'on considère l'ensemble des points de la frontière extérieure $\partial\Omega_e$ comme des points de contrôle, on effectue de l'*inpainting* par diffusion. Or, comme expliqué dans le chapitre 2, les approches d'*inpainting* basées diffusion, si elles sont efficaces sur des petits masques réparties de manière éparses dans l'image, peinent à compléter fidèlement des masques de grande taille.

SAID *et al.* [STB18] se concentrent sur la complétion de tâches spéculaires qui peuvent survenir si la surface d'un plan reflète les rayons lumineux d'une source. Pour cela, ils proposent de modéliser la tâche via une représentation par ellipse. Ils estiment, à partir des isocontours présents dans la zone connue, les ellipses qui les modélisent. Il suffit alors de compléter dans le masque Ω les isocontours à partir des paramètres des ellipses. Nous détaillons cette méthode dans la section suivante.

Bien que cette méthode fonctionne bien pour des tâches spéculaires, le modèle contraint de l'ellipse empêche de l'utiliser pour une carte de luminosité ayant des isocontours qui ne sont pas modélisables par des ellipses. Cependant, nous nous basons dessus et généralisons leur approche (estimation des isocontours puis propagation) pour développer notre propre méthode qui est indépendante de la forme des isocontours. Bien

que leur méthode gère la composante spéculaire, nous pouvons nous en inspirer car les cartes de luminosité peuvent aussi bien représenter les tâches spéculaires que les variations de luminosité dues à la composante diffuse. Nous détaillons notre méthode dans la prochaine sous-section.

5.4.4 Complétion par isocontours

Dans cette section, nous décrivons notre approche de complétion de carte de luminosité. Cette approche se sert des courbes de Hermite pour compléter les isocontours au sein d'un masque Ω. Une fois des isocontours complétés, les valeurs estimées sont propagées par une approche de diffusion classique.

Estimation des isocontours dans la zone connue

À partir de la carte de luminosité $\mathrm{C}_{\mathrm{lum}}$, nous devons créer un ensemble d'isocontours. Pour cela, nous utilisons un histogramme de quantification pour segmenter $\mathrm{C}_{\mathrm{lum}}$ en plusieurs niveaux d'intensité. Nous considérons qu'un pixel p d'intensité $\mathrm{C}_{\mathrm{lum}}(p)$ est sur un isocontour H_z si $\mathrm{C}_{\mathrm{lum}}(p) = z$ et si au moins un pixel de son 4-voisinage $\mathcal{V}_p^4$ a une intensité $\mathrm{C}_{\mathrm{lum}}(p)$ égale à $z-1$. Concernant le nombre d'isocontours à générer, nous estimons de manière empirique quel est le nombre minimum de niveaux d'intensité pour reconstruire la carte $\mathrm{C}_{\mathrm{lum}}$ à partir de ces isocontours. Un nombre trop faible de niveaux ne reconstruit pas fidèlement la carte $\mathrm{C}_{\mathrm{lum}}$. À l'inverse, un nombre trop élevé n'est pas nécessaire pour reconstruire fidèlement la carte de luminosité. La figure 5.7 montre l'influence de ce nombre. On peut y voir qu'un nombre minimal d'isocontours est nécessaire pour pouvoir reconstruire convenablement la carte $\mathrm{C}_{\mathrm{lum}}$. Nous choisissons dans nos applications d'échantillonner sur 32 niveaux d'intensité, répartis de façon régulière entre z_{min} et z_{max}, ce qui fournit un résultat de reconstruction quasi-parfaite pour toutes les cartes de luminosité testées. La prochaine sous-section présente notre proposition pour compléter les isocontours dans le masque Ω.

Détermination des isocontours dans le masque

Maintenant que nous connaissons les isocontours dans la zone connue $O\backslash\Omega$, nous devons estimer leurs parcours dans le masque Ω. Nous commençons par estimer les isocontours selon leur niveau d'intensité. Pour cela, nous notons $\mathcal{I}_k$ l'ensemble des pixels de la frontière extérieure $\partial\Omega_e$ ayant une intensité égale à k :

$$\mathcal{I}_k = \{p \in \partial\Omega_e, \mathrm{C}_{\mathrm{lum}}(p) = k\} \tag{5.20}$$

Tour d'abord, nous supposons Ω fermé et connexe sans trou, donc avec une seule frontière continue ; nous avons de ce fait $\mathrm{card}\,\mathcal{I}_k = 2n$. Comme les isocontours sont continus, tout isocontour qui entre dans le masque est sensé sortir du masque.
Pour estimer le parcours de chaque isocontour, nous devons maintenant déterminer le pixel d'entrée ainsi que le pixel de sortie tout en respectant les propriétés de la carte de luminosité $\mathrm{C}_{\mathrm{lum}}$, notamment le fait que deux isocontours ne doivent pas se croiser. Nous représentons le problème via un graphe circulaire $G = (N, A)$ (voir figure 5.8).

Chaque nœud de N est un élément de $\mathcal{I}_k$. Comme $\delta\Omega$ est fermée, nous pouvons créer une relation de voisinage entre chaque élément de $\mathcal{I}_k$ et le graphe ainsi créé est un cycle simple. Deux points de $\mathcal{I}_k$ sont voisins si on peut aller de l'un à l'autre en parcourant la frontière extérieure sans rencontrer un autre point de $\mathcal{I}_k$. Les arêtes du graphe G représentent cette relation de voisinage. Dans la figure 5.8, le pixel p_1 est voisin avec p_2 et p_6. Par cette représentation, chaque élément de $\mathcal{I}_k$ a exactement 2 voisins.

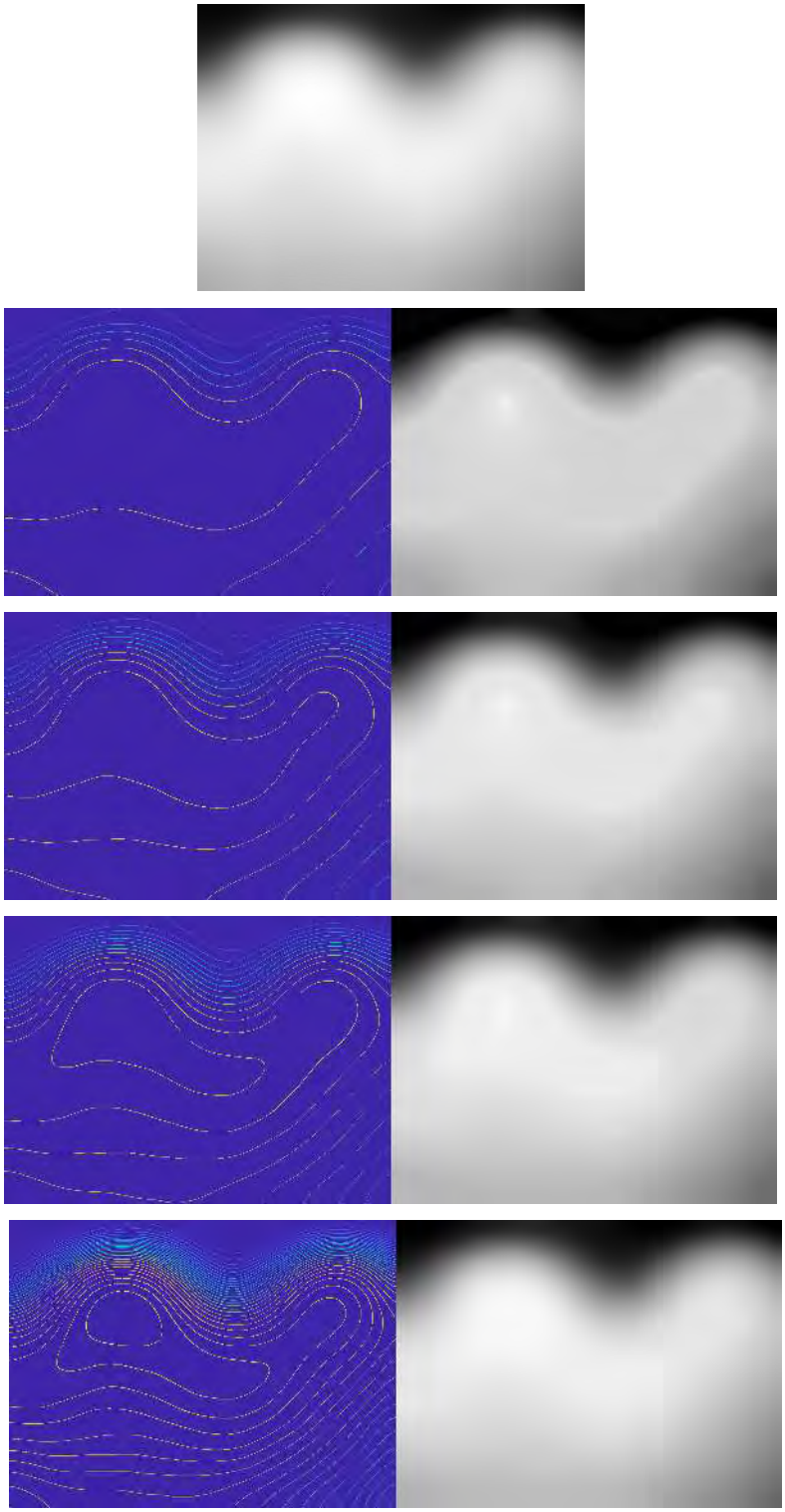


FIGURE 5.7 – Influence du nombre de niveaux d'intensité dans la reconstruction de la carte de luminosité. Première ligne : vérité terrain. Les lignes suivantes montrent la reconstruction de C_{lum} avec $8, 12, 16$ et 32 niveaux d'intensité. On peut constater que pour un faible nombre de niveaux (de la deuxième à la quatrième ligne), des détails de la carte n'ont pas pu être retrouvés. À partir de 32 niveaux d'intensité, la reconstruction est égale à la vérité-terrain.

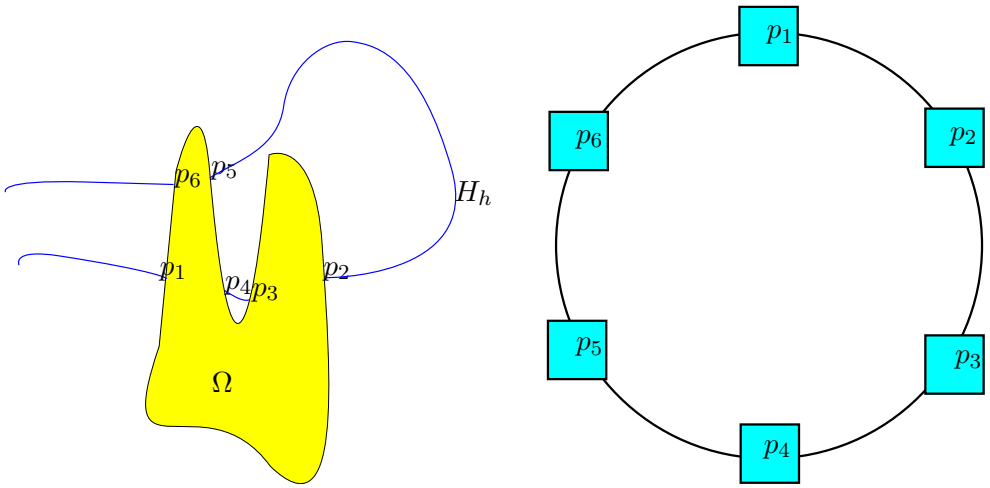


FIGURE 5.8 – Représentation par un graphe circulaire d'isocontours traversant le masque Ω d'une carte de luminosité C_{lum}. Gauche : un isocontour H_h traversant le masque Ω en 6 points. Droite : la représentation par graphe associée.

L'objectif est maintenant d'associer chaque élément p de $\mathcal{I}_k$ avec un autre élément $q \neq p$ de $\mathcal{I}_k$. Cette représentation nous permet d'éliminer les combinaisons, qui créent des intersections entre isocontours. Par exemple, en reprenant l'exemple de la figure 5.8, la combinaison (p_1, p_5) n'est pas possible car l'isocontour entrant dans le masque par p_6 serait obligé de traverser l'isocontour passant par p_1 et p_5. En généralisant, on peut dire qu'une liaison (p_i, p_j) est valable s'il existe un nombre pair de points de $\mathcal{I}_k$ entre les pixels p_i et p_j. La figure 5.9 montre les combinaisons possibles qui respectent la propriété ci-dessus dans le cas où $\operatorname{card} \mathcal{I}_k = 6$.

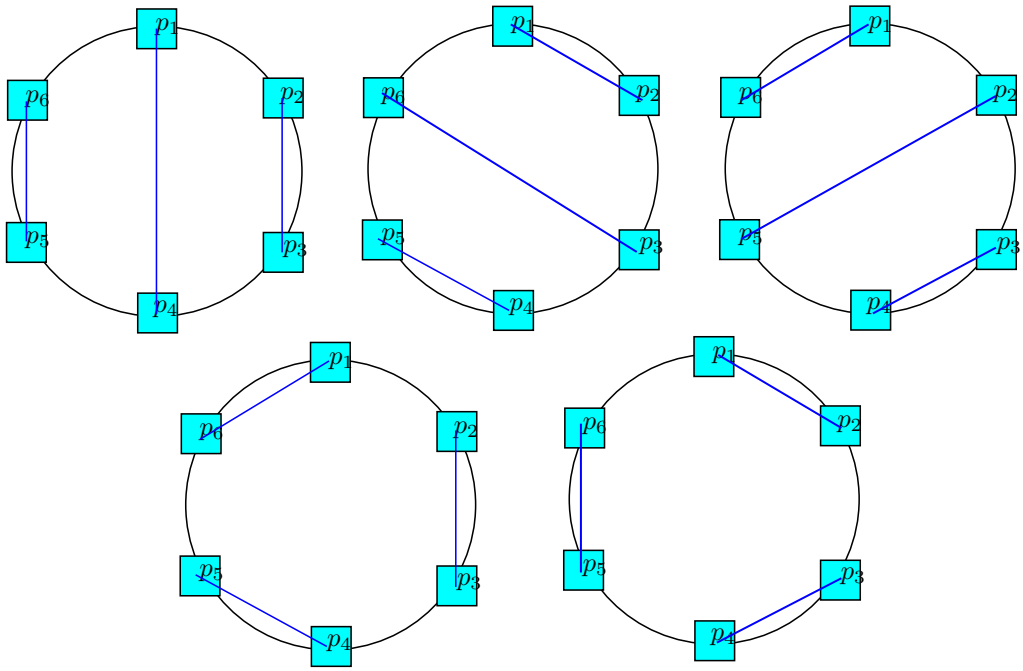


FIGURE 5.9 – Ensemble des combinaisons valables pour le cas à 6 pixels où passe un isocontour, en utilisant la configuration de la figure 5.8.

Malgré cela, il peut y avoir des configurations où effectivement des isocontours

peuvent se croiser en raison de la position des pixels et de leur gradient à la frontière extérieure. Pour sélectionner la meilleure combinaison parmi les possibles, nous minimisons l'énergie suivante :

$$E = \sum_{p_1,p_2 \in \mathcal{I}_k} E_d\left(p_1,p_2\right) + \sum_{p_1,p_2,p_3,p_4 \in \mathcal{I}_k} E_r\left(p_1,p_2,p_3,p_4\right) \tag{5.21}$$

avec

$$E_d\left(p_1,p_2\right) = \text{longueur}(c(p_1,p_2)) \tag{5.22}$$

où longueur$(c(p_1,p_2))$ est la longueur de la courbe représentant l'isocontour et passant par p_1 et p_2 et $E_r\left(p_1,p_2,p_3,p_4\right) = +\infty$ si les courbes générées par (p_1,p_2) et p_3,p_4 se croisent, 0 sinon.

La première énergie favorise les combinaisons qui produisent des isocontours entre des pixels proches. La deuxième élimine les combinaisons où deux isocontours se croisent. Maintenant que nous avons expliqué comment sont sélectionnées les combinaisons, nous décrivons dans la prochaine sous-section comment nous complétons un isocontour passant dans le masque Ω.

Construction par les courbes de Hermite

Soit un isocontour passant dans le masque Ω joignant les points $\mathbf{M}_1$ et $\mathbf{M}_2$ de la frontière $\partial\Omega_e$. Les tangentes normalisées de l'isocontour aux points $\mathbf{M}_1$ et $\mathbf{M}_2$ sont respectivement notées $\vec{\mathbf{m}}_1$ et $\vec{\mathbf{m}}_1$. Pour compléter un isocontour dans le masque Ω, nous utilisons des polynômes cubiques de Hermite. On appelle polynôme cubique de Hermite un polynôme de degré trois qui est le polynôme de degré minimal interpolant une fonction en deux points et leurs tangentes. Nous définissons d'abord l'interpolation de Hermite.

Définition 5.4.1. Cas d'une fonction à une variable

Soit f une fonction de classe C^1 d'une variable définie sur un segment $[a,b]$ et à valeurs réelles. Soient x_1 et x_2 deux points distincts de $[a,b]$. Si $f(x_1) = y_1, f(x_2) = y_2$ et $f'(x_1) = m_1, f'(x_2) = m_2$, alors il existe un unique polynôme $P(X) = aX^3+bX^2+cX+d$ de degré 3 tel que

$$P(x_1) = y_1, \quad P(x_2) = y_2, \quad P'(x_1) = m_1, \quad P'(x_2) = m_2 \tag{5.23}$$

Définition 5.4.2. Cas d'une courbe paramétrée

Soit $c(t) = (x(t), y(t))$ une courbe de classe C^1 de $\mathbb{R}^2$ définie sur un intervalle $[t_1,t_2]$. La courbe c atteint le point $\mathbf{M}_1 = (x_1,y_1)$ (respectivement $\mathbf{M}_2 = (x_2,y_2)$) à $t = t_1$ (respectivement $t = t_2$) avec une tangente égale à $\vec{\mathbf{m}}_1 = (u_1,v_1)$ (respectivement $\vec{\mathbf{m}}_2 = (u_2,v_2)$). Il existe un polynôme $P(X) = a_1X^3 + b_1X^2 + c_1X + d_1$, à valeur dans $\mathbb{R}^2$, de degré 3 tel que :

$$P(t_1) = (x_1,y_1), \quad P(t_2) = (x_2,y_2), \quad P'(t_1) = (u_1,v_1), \quad P'(t_2) = (u_2,v_2) \tag{5.24}$$
$$\tag{5.25}$$

L'avantage des polynômes de Hermite est d'utiliser la valeur de la tangente aux points d'entrée et de sortie pour définir le parcours de l'isocontour.

Cependant, si les points sont connus, nous devons décider de la norme des tangentes (actuellement normalisées) ainsi que de l'intervalle $[t_1,t_2]$.

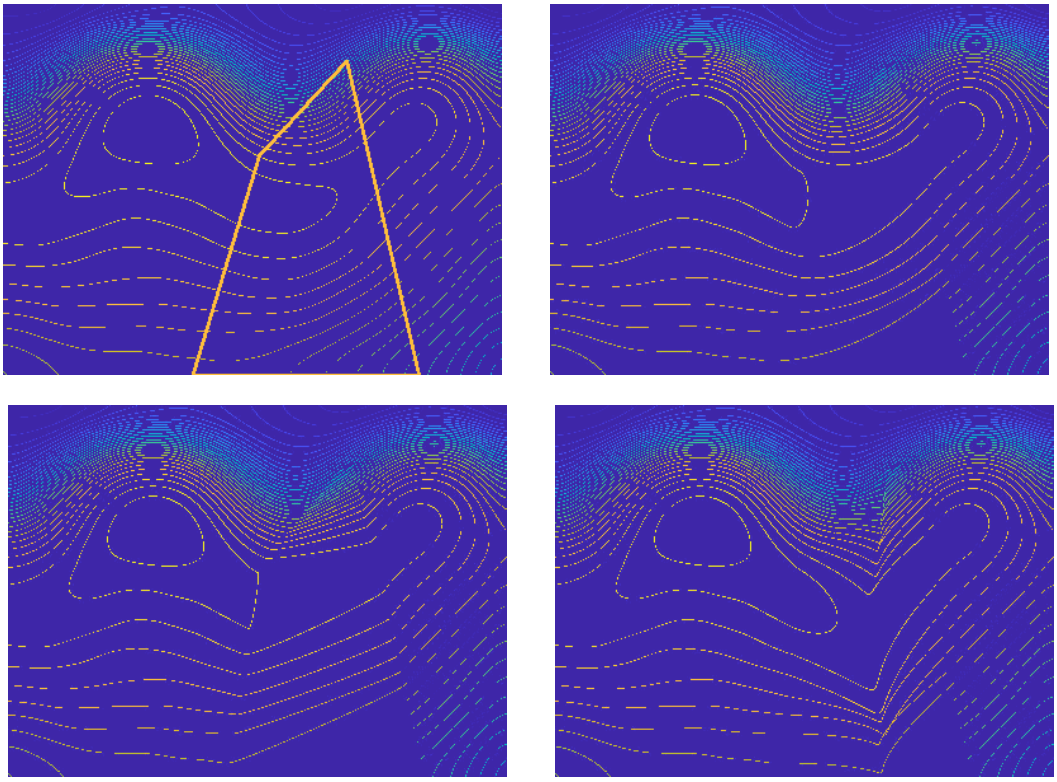


FIGURE 5.10 – Influence de la norme de la tangente dans le calcul des polynômes de Hermite. Une norme faible conduit à des courbes dont l'allure ressemble à des droites (deuxième ligne, image de gauche). À l'inverse, une norme élevée conduit à des problèmes de courbures (deuxième ligne, image de droite). L'image de droite de la première ligne montre un cas intermédiaire où les isocontours ont des courbures attendues.

Choix des normes des tangentes Le choix de la norme des tangentes $\vec{\mathbf{m}}_1$ et $\vec{\mathbf{m}}_2$ aux points $\mathbf{M}_1$ et $\mathbf{M}_2$ où l'isocontour traverse le masque Ω a une influence sur la forme de la courbe comme le montre la figure 5.10. Une valeur faible crée une courbe avec une courbure plus forte aux points $\mathbf{M}_1$ et $\mathbf{M}_2$ et faible à l'intérieur du masque (nous sommes proches d'une droite). À l'inverse, une valeur élevée va créer des courbes avec très "raides", c'est-à-dire avec les courbures très faibles aux points $\mathbf{M}_1$ et $\mathbf{M}_2$ et concentrer la courbure vers les paramètres au centre de l'intervalle créant presque des "cassures". Pour gérer les deux situations, nous proposons de baser la norme sur la distance entre les points d'entrée et de sortie de l'isocontour dans le masque Ω, de ce fait la paramétrisation induite sera cohérente entre les courbes. Cette cohérence entre les différentes courbes est importante car nous souhaitons éviter les intersections entre isocontours.

$$\| \vec{\mathbf{m}_\mathbf{i}} \| = \| \mathbf{M}_2 - \mathbf{M}_1 \|_2 \, . \tag{5.26}$$

Choix de l'intervalle $[t_1, t_2]$ Intuitivement, l'intervalle $[t_1, t_2]$ représente le temps pour parcourir la courbe entre les points $\mathbf{M}_1$ et $\mathbf{M}_2$. Donc des intervalles de même longueur vont créer les mêmes courbes : on peut fixer arbitrairement l'une des extrémités de l'intervalle. Par commodité et sans perte de généralité, nous fixons $t_1 = 0$, nous devons maintenant choisir t_2. La première ligne de la figure 5.11 nous montre l'influence du choix de ce paramètre dans la génération des courbes. Une valeur t_2 proche de $t_1 = 0$ va laisser un temps court pour parcourir la distance entre $\mathbf{M}_1$ et $\mathbf{M}_2$ et donc tendre

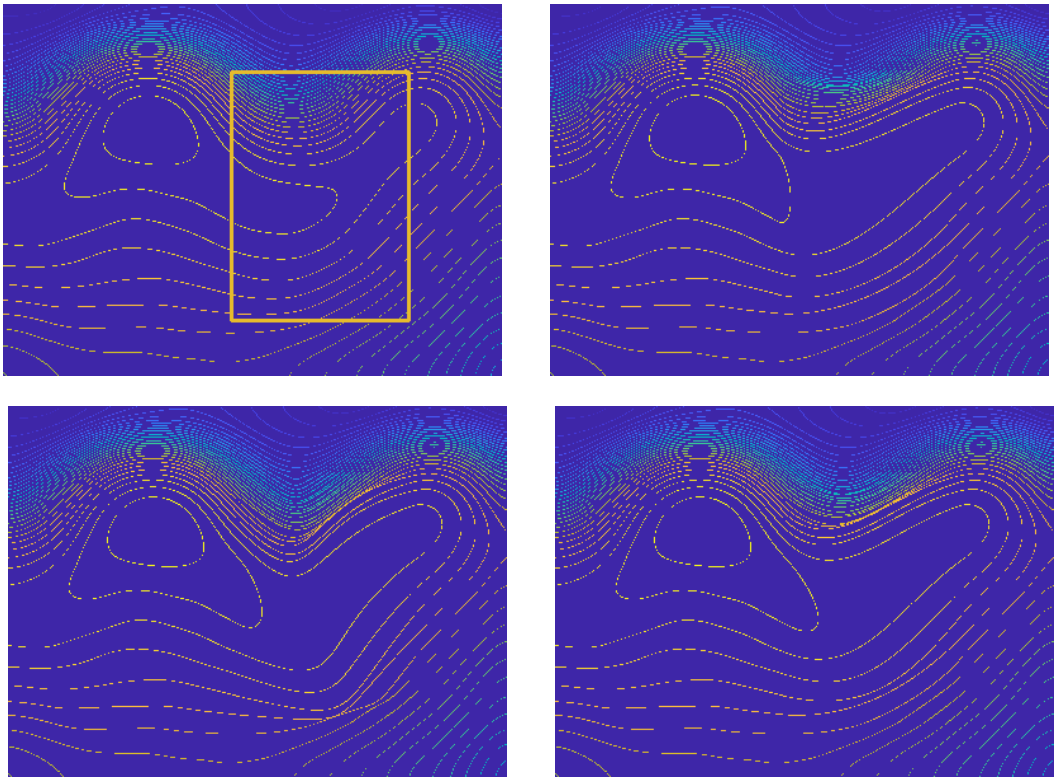


FIGURE 5.11 – Influence de l'intervalle $[t_1, t_2]$ dans le calcul des polynômes de Hermite. Un intervalle de longueur élevée va laisser un temps long et créer des courbes plus amples ce qui est favorable pour les courbes qui ont des tangentes d'entrée et de sortie de directions opposées (première ligne, image de droite). À l'inverse, un intervalle ayant une longueur faible va tendre les courbes (car elles ont moins de temps pour parcourir la distance entre les deux points) ; ce qui est attendu pour les courbes dont les tangentes partagent la même direction (deuxième ligne, image de gauche). L'image de la deuxième ligne montre notre proposition de gestion dynamique de l'intervalle en fonction du produit scalaire (*cf.* équation 5.27) entre les deux tangentes (ici avec $\alpha = 1.5$). Notre résultat est celui qui se rapproche au mieux de la vérité-terrain (première ligne, image de gauche, le masque Ω est encadré en jaune).

les isocontours ; ceci est souhaitable si les tangentes d'entrée et de sortie $\overrightarrow{\mathbf{m}}_2$ et $\overrightarrow{\mathbf{m}}_2$ sont proches. Par contre, les courbes dont les tangentes d'entrée et de sortie ont des directions opposées vont générer une courbe en boucle, et donc doivent pouvoir répartir la courbure le long d'une courbe plus longue. De ce fait, une valeur t_2 plus élevée va créer des courbes plus amples, de forme attendue pour celles qui des tangentes d'entrée et de sortie ayant des directions opposées. Par contre, les courbes, dont les tangentes d'entrée et de sortie ont la même direction, vont avoir une courbe en boucle alors qu'on attend une courbure plus faible. Pour gérer de manière dynamique les deux situations, nous proposons de calculer le terme t_2 en utilisant le produit scalaire entre les deux tangentes :

$$t_2 = \frac{\| \overrightarrow{\mathbf{M_1 M_2}} \|_{\frac{1}{2}}}{\| \overrightarrow{\mathbf{m_i}} \|_2} - \alpha \, (\overrightarrow{\mathbf{m_1}} | \overrightarrow{\mathbf{m_2}}) \tag{5.27}$$

où $\| \cdot \|_2$ est la norme euclidienne, est la distance centripète [YSK11] et α est un paramètre pour contrôler l'influence du produit scalaire dans la génération des courbes. L'image de gauche de la deuxième ligne de la figure 5.11 montre un exemple de complétion d'isocontour avec un intervalle $[t_1, t_2]$ variant en fonction de l'équation 5.27. On peut constater que les deux types d'isocontours sont gérés, ce qui n'est pas le cas des images de la première ligne.

Cas d'un masque ouvert

Nous avons considéré précédemment que le masque Ω était fermé. Cependant, il est possible que celui-ci soit ouvert. En effet, dans notre contexte de Réalité Diminuée, les objets à effacer sont souvent collés aux murs. Cela génère, dans les images rectifiées, des masques qui se situent aux bords. Dans cette situation, il est probable qu'un isocontour qui entre dans le masque, sorte par la frontière intérieure du masque qui est située aux bords. Or, n'avons donc pas de frontière extérieure dans cette partie de l'image, c'est-à-dire, que nous n'avons pas d'information utilisable pour construire les isocontours.
Nous devons donc imaginer le point de sortie d'un isocontour donné ainsi que sa tangente en ce point. Pour cela, nous proposons de faire une diffusion anisotrope [TD05] uniquement sur la zone du masque Ω qui se situe au bord de l'image afin d'avoir une estimation qui respecte la continuité de la carte de luminosité. Il suffit après, pour un niveau d'intensité donné, d'ajouter les points des bords de l'image au graphe défini dans la section 5.4.4. La figure 5.12 montre deux cas où le masque est situé aux bords de l'image. On peut constater la diffusion fournit une estimation correcte de la position des points de sortie ainsi que de leur tangente.

Génération de la carte de luminosité

Une fois les courbes des isocontours reconstruites dans le masque Ω, on peut générer simplement la carte de luminosité par une approche de diffusion [TD05]. Auparavant, si notre méthode de sélection via un graphe nous garantit que les isocontours de même intensité ne se croisent, cela n'est pas le cas si les isocontours ne sont pas de même intensité. Nous devons nous assurer que les isocontours d'intensité différente ne se croisent pas. Cette régularisation est abordée dans les perspectives. Un flou gaussien est finalement appliqué pour lisser les rares isocontours qui auraient été mal reconstruites.

5.4.5 Résultats

Dans cette sous-section, nous présentons des résultats de notre méthode, qui est comparée avec l'approche par point de contrôle de Siltanen [Sil15] et par une approche

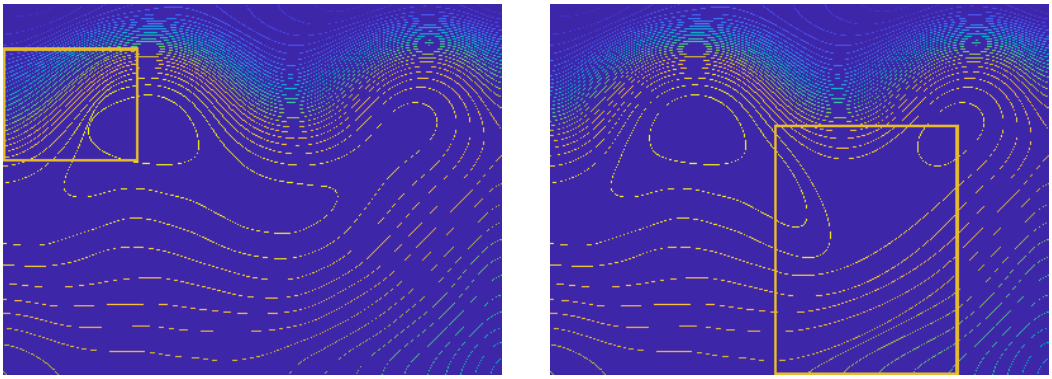


FIGURE 5.12 – Construction d'isocontours dans des cas où le masque est aux bords de l'image (encadré en jaune). Le fait d'initialiser avec une diffusion le long du bord de l'image permet d'avoir une bonne estimation pour compléter chaque isocontour dans le masque Ω.

classique de diffusion [CS01b].
Nous comparons d'abord les résultats concernant la complétion des isocontours. Nous comparons ensuite les cartes de luminosité complétées des trois approches. Enfin, nous comparons le rendu final avec la carte de texture associée. Dans ce dernier cas, nous utilisons notre approche d'*inpainting* expliquée dans le chapitre 3.

Comparaisons des isocontours

Nous comparons dans les figures 5.13 et 5.14 les isocontours reconstruites dans un masque Ω. Nous pouvons constater que notre méthode fournit un meilleur respect de la trajectoire des isocontours que les deux autres méthodes, notamment au centre du masque et au bord du bas de l'image.

Comparaisons des cartes C_{lum}

Nous comparons les cartes de luminosité C_{lum} à partir des situations des figures 5.13 et 5.14. Les figures 5.15 et 5.16 montrent les cartes complétées. Nous pouvons constater globalement que notre méthode arrive à compléter convenablement des masques de grande taille dans des cartes de luminosité. Le fait de guider la diffusion via les isocontours construits via les polynômes de Hermite fournit un meilleur résultat qu'une diffusion classique ou qu'une propagation par points de contrôle.

Comparaison d'images rectifiées (avec texture)

Nous comparons ici le rendu final d'une image rectifiée. Pour rappel, celle-ci est divisée en deux images : une carte de texture et la carte de luminosité. La première est complétée avec la méthode d'*inpainting* du chapitre 3. La variation de la résolution est contrôlée avec le critère de confiance du chapitre 4.
Nous pouvons constater que notre méthode permet d'avoir un rendu bien plus réaliste qu'avec une propagation classique. Dans la figure 5.17 par exemple, nous pouvons apercevoir une tâche d'ombre au centre du masque (image de gauche), ce qui n'est pas le cas avec notre méthode (image de droite). Comme la carte de luminosité peut aussi représenter la composante spéculaire, nous avons testé sur une texture non mate (*cf*. figure 5.18). Notre méthode fournit également de bons résultats par rapport aux autres méthodes,

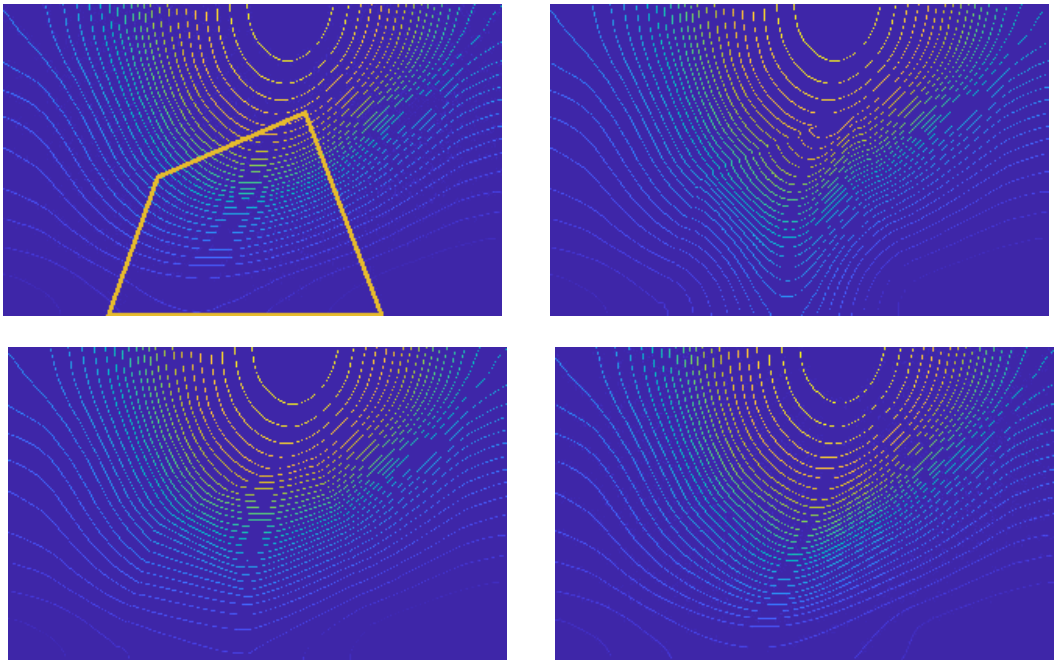


FIGURE 5.13 – Isocontours complétés dans Ω avec différentes méthodes. Première ligne : image de gauche : vérité-terrain avec le masque encadré en jaune ; image de droite : méthode des points de contrôle de SILTANEN [Sil15]. Deuxième ligne : image de gauche, méthode par une propagation basée diffusion [CS01b] ; image de droite : notre méthode par reconstruction des isocontours avec les polynômes de Hermite. On peut constater des problèmes de discontinuités à la frontière dans le résultat de SILTANEN [Sil15]. De plus, le résultat aux bords est très éloigné de la vérité-terrain. L'approche diffusion et la nôtre arrivent à respecter le trajet des isocontours par rapport à la vérité-terrain. Notons que nos isocontours sont plus lisses que ceux de la diffusion.

notamment en terme de propagation de l'intensité. Les deux premières approches (première ligne, image de droite et deuxième ligne, image de gauche) vont propager plus loin de prévu les valeurs élevées d'intensité lumineuse alors que notre méthode (deuxième ligne, image de gauche) contrôle mieux cette propagation.

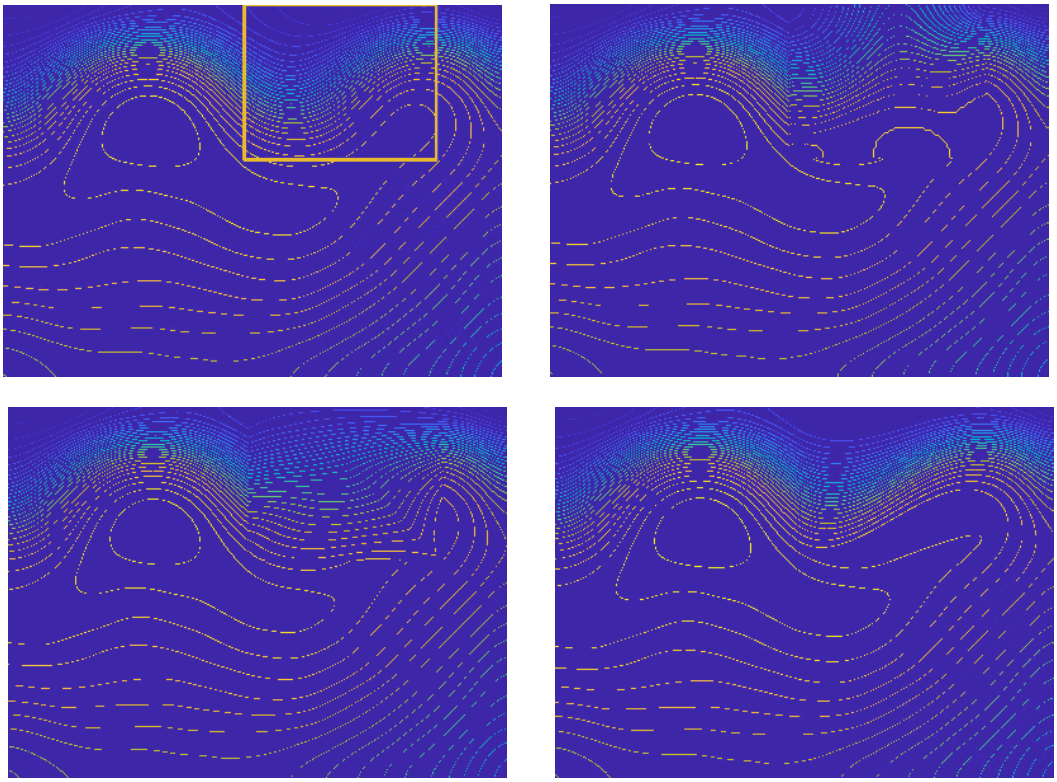


FIGURE 5.14 – Isocontours complétés dans Ω avec différentes méthodes. Première ligne : image de gauche : vérité-terrain avec le masque encadré en jaune ; image de droite : méthode des points de contrôle de SILTANEN [Sil15]. Deuxième ligne : image de gauche, méthode par une propagation basée diffusion [CS01b] ; image de droite : notre méthode par reconstruction des isocontours avec les polynômes de Hermite. Comme dans la figure 5.13, nous observons des problèmes aux bords pour les approches par point de contrôle [Sil15] et par diffusion [CS01b]. Notons aussi les problèmes de propagation dans le masque. En construisant les isocontours d'abord, notre méthode permet de mieux gérer la structure dans les grands masques.

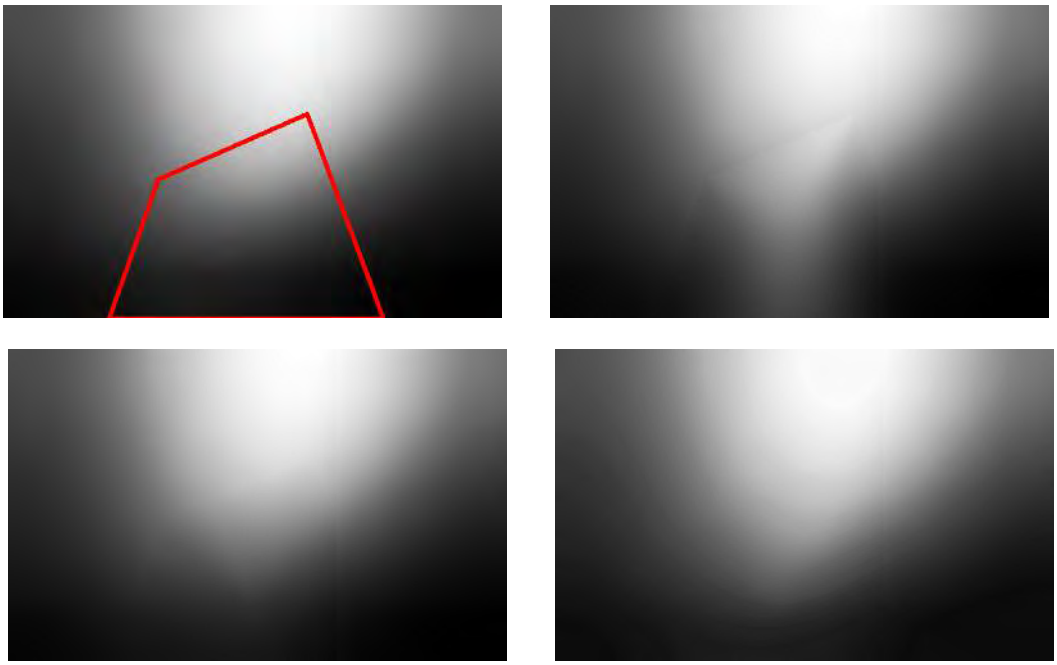


FIGURE 5.15 – Isocontours complétés dans Ω avec différentes méthodes. Première ligne : image de gauche : vérité-terrain avec le masque encadré en rouge ; image de droite : méthode des points de contrôle de SILTANEN [Sil15]. Deuxième ligne : image de gauche, méthode par une propagation basée diffusion [CS01b] ; image de droite : notre méthode par reconstruction des isocontours avec les polynômes de Hermite.

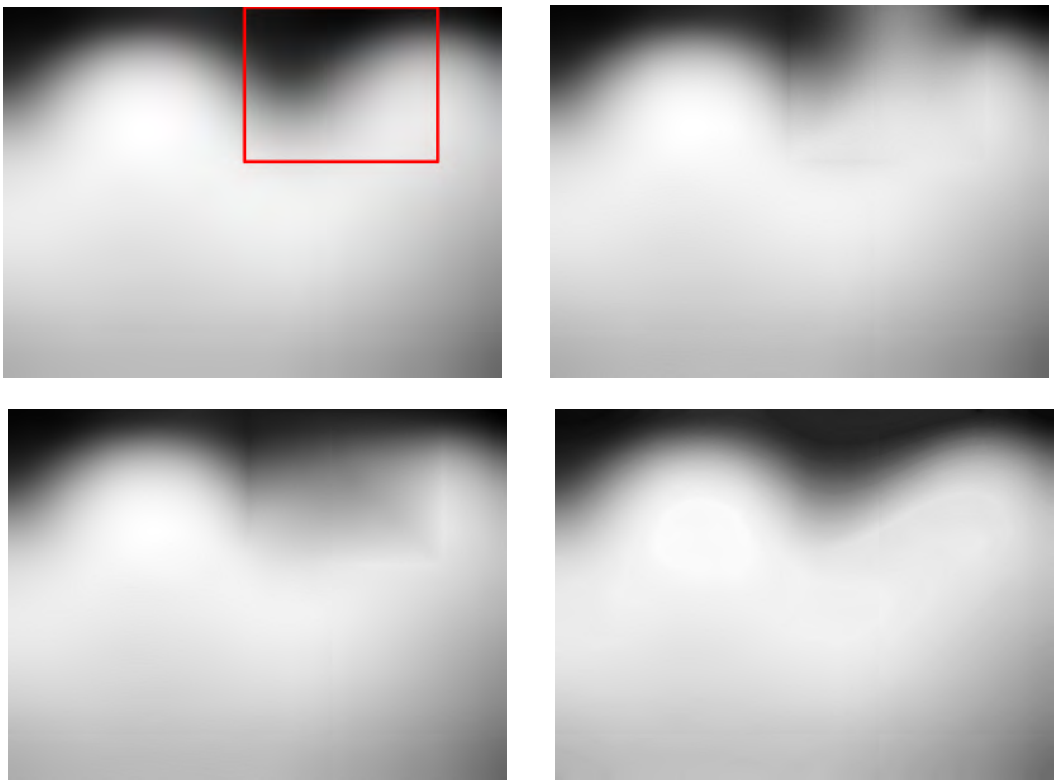


FIGURE 5.16 – Isocontours complétés dans Ω avec différentes méthodes. Première ligne : image de gauche : vérité-terrain avec le masque encadré en rouge ; image de droite : méthode des points de contrôle de SILTANEN [Sil15]. Deuxième ligne : image de gauche, méthode par une propagation basée diffusion [CS01b] ; image de droite : notre méthode par reconstruction des isocontours avec les polynômes de Hermite.

FIGURE 5.17 – Application d'une carte de luminosité à une texture Gauche : avec une méthode par diffusion [CS01a]. Droite : avec notre méthode de reconstruction des isocontours. On peut constater sur l'image de gauche une ombre au milieu du masque, ce qui casse le réalisme du rendu. À l'inverse, notre méthode fournit un rendu bien plus réaliste car la luminosité a été bien mieux propagée.

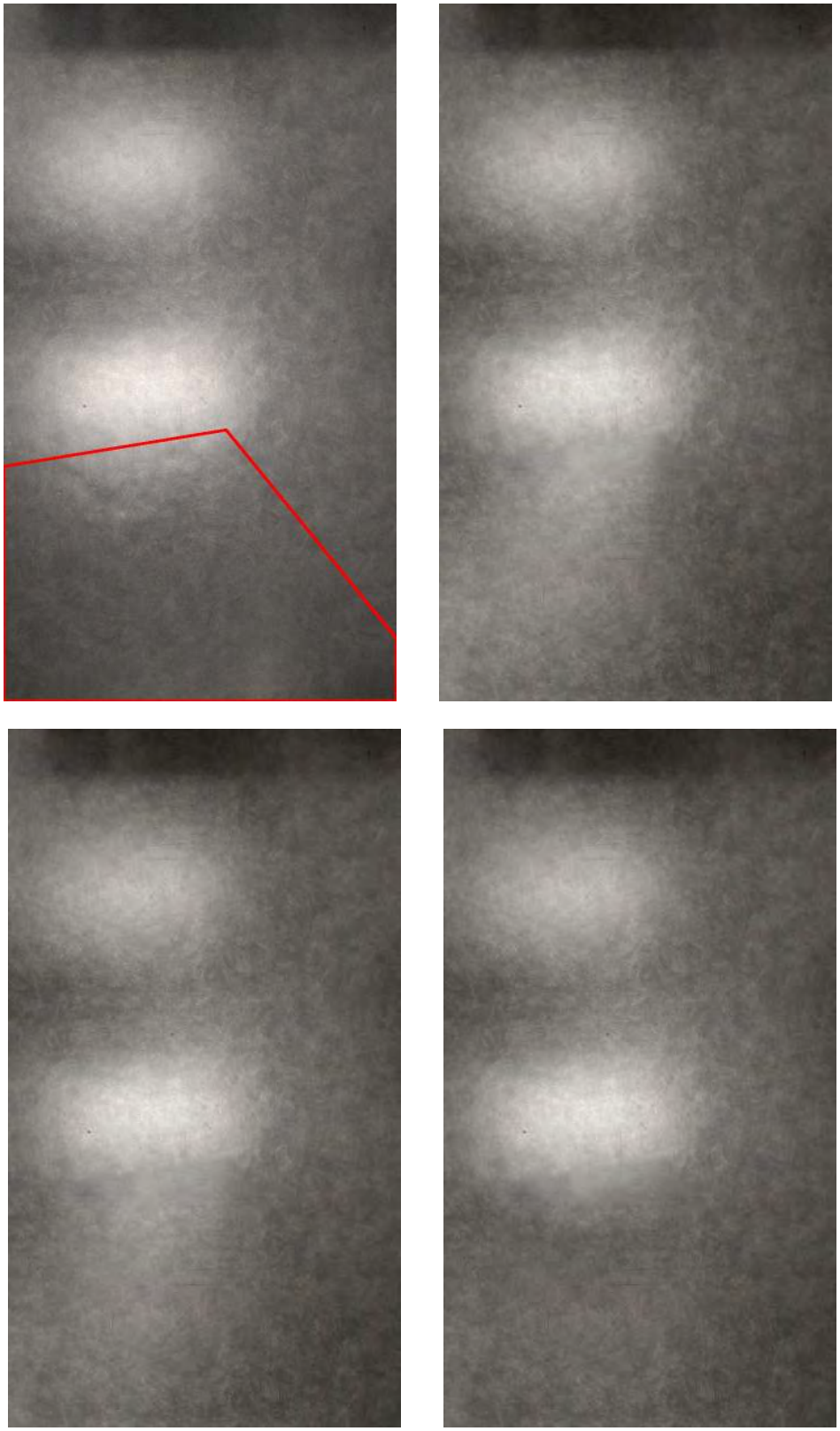


FIGURE 5.18 – Comparaison des méthodes de complétion sur une image fronto-parallèle non mate. En haut, à gauche : vérité-terrain (masque encadré en rouge). En haut, à droite : avec la méthode des points de contrôle [Sil15]. En bas, à gauche : avec une méthode par diffusion [CS01a]. En bas, à droite : avec notre méthode de reconstruction des isocontours. Les deux premières méthodes propagent plus loin qu'attendu les valeurs d'intensité les plus élevées de la tâche spéculaire du bas. À l'inverse, notre méthode contrôle mieux cette propagation.

5.5 Gestion des tâches spéculaires et perspectives

Après avoir abordé la gestion de la luminosité ambiante et de la composante diffuse, nous traitons dans cette section de la gestion de la composante spéculaire. Cette composante n'a pas été approfondie durant la durée de la thèse car nous supposions que la surface était une surface de Lambert. Nous proposons une extension possible en intégrant deux approches qui traitent du cas spéculaire. Premièrement, nous détaillons une méthode [MT14] permettant de détecter les tâches spéculaires dans une image. Deuxièmement, nous expliquons la méthode [STB18] évoquée dans la section précédente pour compléter les tâches spéculaires dans une image.

5.5.1 Distinction entre les composantes diffuse et spéculaire

Comme mentionné dans la section 5.4, la carte de luminosité peut aussi bien représenter l'illumination venant de la composante diffuse que celle venant de la composante spéculaire. Notre méthode peut donc aussi bien compléter une zone masquée d'une carte $\mathrm{C_{lum}}$ qui représente la composante spéculaire d'une image rectifiée I_Δ. Cependant, comme énoncé en introduction, les tâches spéculaires varient en fonction de la position de l'observateur. On peut confondre les deux composantes (spéculaire et diffuse) dans le cas d'une image rectifiée construite à partir d'un seul point de vue. Par contre, il est nécessaire de les traiter de manière différente dans le cas où l'image rectifiée est obtenue à partir de plusieurs points de vue. La normalisation (*cf.* équation 5.11) permet de séparer la texture, représentée par la carte de texture T_Δ, de la luminosité stockée dans $\mathrm{C_{lum\,\Delta}}$. Il faut donc encore dissocier la composante spéculaire de la composante diffuse dans $\mathrm{C_{lum\,\Delta}}$. Nous proposons pour cela d'utiliser deux méthodes de l'état de l'art :

1. Une première méthode [MT14] qui détecte dans une image les tâches spéculaires présentes.
2. Une deuxième méthode [STB18] qui complète les tâches spéculaires cachées en partie par un masque Ω présent dans l'image.

5.5.2 Détection des tâches spéculaires

Morgand et Tamaazousti [MT14] proposent, pour détecter les tâches spéculaires, de définir un seuillage dynamique. Celui-ci est appliqué dans l'espace de couleur HSV car cela permet de mieux représenter la sensibilité de l'œil humain à la lumière [OT06]. De plus, les tâches spéculaires sont caractérisées, dans l'espace HSV, par une saturation faible S et une valeur V élevée.
Tout d'abord, ils effectuent un pré-traitement dans le cas où une image est saturée. Pour cela, ils calculent, une valeur appelée « brillance » B associée à l'image. Celle-ci est comparée à un seuil de brillance T_b. Si $B > T_b$, le contraste est réduit sur l'ensemble de l'image de manière progressive jusqu'à ce que la nouvelle valeur de B soit inférieure au seuil T_b.
Ensuite, ils calculent le seuil idéal T_v selon la valeur V à partir de la brillance B sachant que empiriquement, le seuil idéal T_v est deux fois supérieur à B. Quant au seuil de saturation T_s, il est laissé constant, quelque soit l'image considérée. Un pixel p fera partie d'une tâche spéculaire si :

$$V(p) > T_v \quad \text{et} \quad S(p) < T_s. \tag{5.28}$$

Les pixels sont ensuite regroupés [Sb85] pour former des régions qui représentent des tâches spéculaires possibles. Enfin, un post-traitement est effectué pour éliminer les faux

positifs détectés dans le cas d'images saturées. Pour cela, ils font varier les deux seuils T_s et et T_v sur les régions détectées. Une région sera considérée comme spéculaire si la variation de l'aire est constante par rapport à la variation des deux seuils.
La carte de luminosité $C_{\text{lum}\Delta}$ peut maintenant être considérée comme la somme des images suivantes (ce qui est possible d'après le modèle de PHONG [Pho75]) :

$$C_{\text{lum}\Delta} = C_{\text{lum}\Delta,d} + C_{\text{lum}\Delta,s} \tag{5.29}$$

où $C_{\text{lum}\Delta,d}$ est la carte de la composante diffuse et $C_{\text{lum}\Delta,s}$ est la carte de la composante spéculaire. La carte $C_{\text{lum}\Delta,s}$ est obtenue en récupérant les valeurs des pixels appartenant aux régions détectées précédemment. La carte diffuse $C_{\text{lum}\Delta,d}$ est obtenue en complétant les zones considérées comme des tâches spéculaires dans $C_{\text{lum}\Delta}$ (elles font partie dorénavant du masque Ω). Les parties du masque Ω dans $C_{\text{lum}\Delta,s}$ sont complétées avec la méthode de la sous-section suivante.

5.5.3 Complétion des tâches spéculaires

SAID *et al.* [STB18] se concentrent sur la complétion de tâches spéculaires qui peuvent survenir si la surface d'un plan reflète les rayons lumineux d'une source. Pour cela, ils proposent de modéliser la tâche spéculaire via une représentation par ellipse. Ils partent du principe qu'une tâche spéculaire génère des isocontours qui sont concentriques. Les isocontours peuvent donc être modélisés par des ellipses. Ils estiment donc, à partir des isocontours présents dans la zone connue, les caractéristiques des ellipses qui les modélisent. Or, comme dans notre méthode, les isocontours estimés dans le masque peuvent se croiser. Pour contrer cela, SAID *et al.* [STB18] se servent du fait que les isocontours sont concentriques dans le cas spéculaire. Les ellipses qui ne respectent pas cette propriété sont retirées du processus. Ensuite, les pixels passant par une ellipse obtiennent la valeur de l'isocontour associé. À partir de ces estimations, ils propagent dans la zone masquée les valeurs reconstruites en utilisant les splines *Thin-Plate* comme moyen d'interpolation.
La figure 5.19 montre un exemple de propagation de la luminosité dans le cas spéculaire. À partir d'une zone connue entourant le masque Ω (images de gauche), chaque isocontour est estimé par une ellipse. Cette estimation permet alors de compléter les pixels qui font partie de cette ellipse (images du milieu). Il est alors possible de compléter la luminosité à l'intérieur du masque (images de droite).

5.5.4 Cas de la régularisation

Comme évoqué dans la section 5.4, il faut s'assurer que les isocontours que nous avons complétés ne se croisent pas. À l'inverse de SAID *et al.* [STB18], nous n'avons pas l'hypothèse de concentricité (il peut y avoir des "cols" dans la carte C_{lum}). Une idée naïve est de ne pas tenir compte de ces isocontours pour générer la carte de luminosité. Cependant, une idée plus intéressante est de faire varier *a posteriori* les intervalles de temps et les normes des tangentes de chaque isocontour en minimisant par exemple une énergie qui pénaliserait les courbes qui se croisent.

5.6 Conclusion

Dans ce chapitre, nous avons abordé le cas de la luminosité dans le cadre d'un scénario de Réalité Diminuée. Nous avons rappelé les trois composantes de la littérature [Pho75] qui définissent la luminosité dans un environnement. Pour chacune des trois

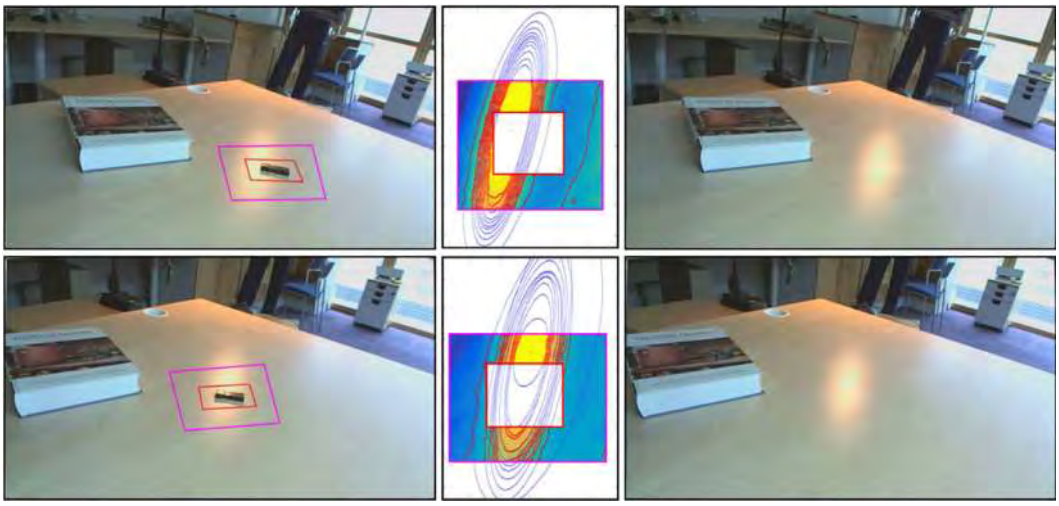


FIGURE 5.19 – Propagation de la luminosité par estimation des isocontours via une représentation par ellipse des tâches spéculaires (figure extraite de SAID *et al.* [STB18]). Chaque ligne correspond à une image issue d'une *frame*. Colonne de gauche : image d'entrée, la zone encadrée en rouge est la zone masquée, la zone encadrée en violet est la zone d'estimation des isocontours. Colonne du milieu : chaque isocontour est défini par une ellipse ; il est possible alors, pour chaque isocontour, d'interpoler son passage dans la zone masquée. La propagation de la luminosité est ensuite guidée par ces isocontours. Colonne de droite : rendu final.

composantes, nous avons utilisé des méthodes existantes adaptées à notre problème, notamment dans le cas de l'illumination ambiante et de la composante spéculaire.
Pour la composante diffuse, nous avons proposé une amélioration des méthodes courantes pour compléter la carte de luminosité qui représente l'illumination diffuse. Pour cela, nous avons d'abord construit les isocontours qui traversent le masque en utilisant les polynômes de Hermite. À partir de là, il est possible de générer efficacement la carte de luminosité dans la zone masquée.

Dans les prochains chapitres, nous aborderons les applications que permettent les différents chapitres de cette partie. Nous parlerons notamment dans le chapitre 6 du cas où nous avons plusieurs vues à projeter dans l'image rectifiée. La problématique sera de décider notamment quelle donnée copier si des pixels de plusieurs vues sont projetées au même endroit de l'image.
Nous parlerons également dans le chapitre 7 de l'extension à la géométrie non planaire à travers la texturisation de modèles représentés par des surfaces canaux.
Enfin, nous présenterons dans le chapitre 8 une application temps-réel de Réalité Diminuée qui fait la synthèse de toutes les approches utilisées et développées durant cette thèse.

Troisième partie

Applications

Chapitre 6

Scénarios de Réalité Diminuée à partir d'une ou plusieurs images

Sommaire

Dans ce chapitre, nous allons aborder deux scénarios de Réalité Diminuée qui se basent sur les notions abordées dans les précédents chapitres. Ces scénarios sont illustrés à travers deux applications de preuve de concept chez Innersense. Faisant partie de l'équipe R&D d'Innersense, j'ai beaucoup participé à ces intégrations.

6.1 Scénario à partir d'une image : l 'application « feuille blanche »

L'application « feuille blanche » est la première application interactive développée par le pôle R&D d'Innersense où les travaux de cette thèse ont été intégrés. Réalisée en 2017, le critère de confiance (*cf*. chapitre 4) était la seule contribution développée et intégrée dans l'application. Elle avait pour objectif d'effectuer un scénario de Réalité Diminuée basée sur une unique image dans laquelle on voit une page blanche A4. L'idée est d'utiliser les dimensions connues de la feuille blanche pour la considérer comme un marqueur de Réalité Augmentée. L'avantage de cette application est de n'avoir besoin que d'une seule image, une contrainte forte est d'apercevoir clairement une feuille blanche A4.

Pour détecter la feuille A4, l'utilisateur est invité à sélectionner le centre de la feuille dans l'image. Les coins sont ensuite automatiquement calculés à partir de ce centre. Pour cela, nous nous basons sur le fait que la feuille soit blanche pour appliquer un seuillage sur l'image afin de ne garder que les valeurs les plus élevées. Ensuite, nous effectuons un agrandissement de région (ou *region growing*) sur l'image binaire générée à partir du seuillage. Pour rappel, le principe du *region growing* est d'étendre la zone autour de

pixels clefs appelés *seed points* à partir d'un critère de sélection utilisé pour comparer les pixels autour des *seed points*. Si un pixel voisin à un *seed point* valide le critère de sélection, il devient à son tour un point clef, ou *seed point*. La zone s'agrandit donc, de manière itérative, en comparant la frontière extérieure avec la frontière intérieure de la zone. Dans notre cas, le centre de la feuille sélectionné par l'utilisateur est notre *seed point* tandis que le critère de sélection est le fait de partager la même intensité (qui vaut ici 1 au pixel du centre de la feuille). Enfin, nous sélectionnons les pixels représentant les coins de la feuille en adaptant le critère $C(p)$ de Criminisi *et al.* [CPT03] et explicité dans l'équation 2.19 du chapitre 2. L'idée du critère est la suivante : les pixels des coins de la feuille ont moins de pixels voisins qui appartiennent à la région générée précédemment que les autre pixels de la feuille. Ils auront donc une valeur $C(p)$ plus faible que celle des autres pixels. Nous sélectionnons donc les quatre pixels dont le critère $C(p)$ est le plus faible.

Une fois les coins calculés, nous pouvons calculer l'homographie `H` associant les coins visibles dans l'image et leurs coordonnées dans le plan du sol. La matrice `K` des paramètres intrinsèques est supposée connue puisqu'elle est fournie directement par Android ou iOS. Comme expliqué dans le chapitre 4, nous pouvons en déduire la matrice $[\mathtt{R}|\mathtt{T}]$ des paramètres extrinsèques de la caméra.

La position de la caméra étant maintenant connue, les plans des murs sont déterminés à partir d'un tracé interactif de lignes par l'utilisateur sur la frontière séparant les deux plans dans l'image rendue.
Les zones de chaque plan sont ensuite rectifiées selon l'homographie appropriée. Chaque image rectifiée voit sa luminosité normalisée (*cf.* chapitre 5). La zone cachée par le masque était complétée par une approche d'*inpainting* couplé au critère de confiance défini au chapitre chapitre 4. Au moment où l'application était développée (les contributions du chapitre 3 n'étaient pas encore implémentées), l'utilisateur était invité à choisir l'algorithme d'*inpainting* adapté à la texture. La figure 6.1 montre un exemple de rendu de cette application.

6.2 Scénario à partir de plusieurs images : *RemoveMyKitchen*

Alors que l'application précédente ne permettait qu'une interaction statique (une seule image), *RemoveMyKitchen* propose d'effectuer un scénario de Réalité Diminuée permettant d'obtenir un modèle complet d'une pièce vidée de tous ses éléments (sauf les portes et les fenêtres). *RemoveMyKitchen* se base notamment sur les travaux la thèse de Gohard [Goh18] sur la reconstruction d'intérieur afin d'obtenir la structure d'une pièce simple à partir d'un ensemble de prises de vue ainsi que les poses caméras associées.

6.2.1 Reconstruction de la pièce

Nous abordons dans cette sous-section la reconstruction de la pièce à partir de plusieurs images prises par l'utilisateur. Les images enregistrées servent à la reconstruction mais aussi à la diminution expliquée dans la prochaine sous-section. Nous expliquons de manière succincte la première étape.

Tout d'abord, l'utilisateur est invité à prendre plusieurs photos de la pièce. Les photos doivent former un ensemble qui couvre toute la pièce. Cependant, il est nécessaire d'apercevoir sur chaque photo prise par l'utilisateur un coin à l'intersection entre trois

FIGURE 6.1 – Effacement d'une pièce à partir d'une seule image via l'application « feuille blanche ». Le plan du sol est complété via un *inpainting* basé *PatchMatch* avec critère de confiance. Le plan du mur de droite est complété via une synthèse et un recalage (expliqué dans la suite du chapitre). Le plan du mur de droite est complété via un *inpainting* basé *PatchMatch*. La fenêtre est et traitée à part (sélection manuelle par l'utilisateur) et complétée via un *inpainting* basé *offsets*.

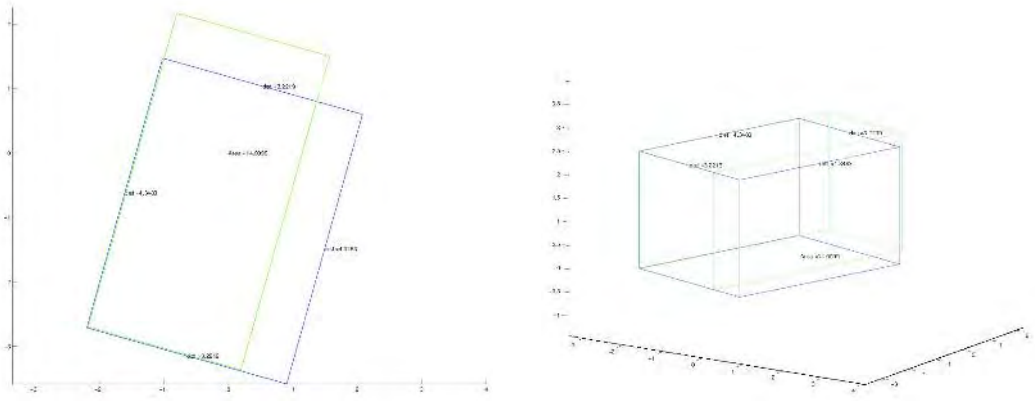


FIGURE 6.2 – Reconstruction d'une pièce à partir des données fournies par l'utilisateur. Le tracé vert représente la reconstruction avant optimisation tandis que le tracé bleu représente la reconstruction après optimisation (figure extraite de GOHARD [Goh18]).

plans de la pièce (généralement le plan du sol et deux plans du mur). Celui-ci doit vraiment être visible. Sur chaque image, les droites correspondant aux intersections entres deux murs voisins sont également fournies par l'utilisateur (dessinées de façon interactive). La connaissance des coins et des droites permet d'avoir une première estimation de la géométrie de la pièce[1]. La résolution se fait via g2o [Küm+11]. La figure 6.2 nous montre la représentation géométrique d'une pièce avant et après optimisation.
Nous demandons à l'utilisateur de pointer, pour chaque plan, le centre d'une zone contenant une tuile répétitive d'une texture à structure régulière pour pouvoir appliquer une synthèse à partir de cette zone si nous manquons d'information pour effectuer une complétion par *inpainting* dans ce plan. Enfin, l'utilisateur est invité à encadrer, dans chaque image, les objets à effacer.
Nous nous retrouvons avec à ce stade avec les données suivantes :

- l'ensemble de prises de vue utilisées durant la reconstruction de la pièce,
- les poses caméra associées à chaque prise de vue,
- les plans délimités de la pièce,
- les masques des objets à supprimer (définis par l'utilisateur),
- les zones de textures à synthétiser.

6.2.2 Diminution de la scène

Nous détaillons dans cette section la brique de diminution de la scène à partir des données obtenues précédemment. La figure 6.3 montre les différentes étapes de ce processus.

Dans la suite du chapitre, nous utilisons la convention du la figure 6.4 concernant le repère caméra : l'axe $\overrightarrow{\mathbf{z}}$ part du centre $\mathbf{C}$ de la caméra pour aller vers l'arrière de la caméra.

Calcul de l'homographie

Pour rappel, nous calculons, pour chaque plan, une homographie pour éliminer les distorsions dues à la perspective. Nous définissons une image I_Δ rectifiée correspondant au plan Δ. Celle-ci représente le plan Δ vu de face. Pour chaque prise de vue I_i, nous calculons plus particulièrement l'homographie $\mathtt{H}_i^\Delta$ qui projette la zone appartenant au

1. Si le lecteur veut plus d'informations, il est invité à lire le chapitre 5 de la thèse de Gohard [Goh18].

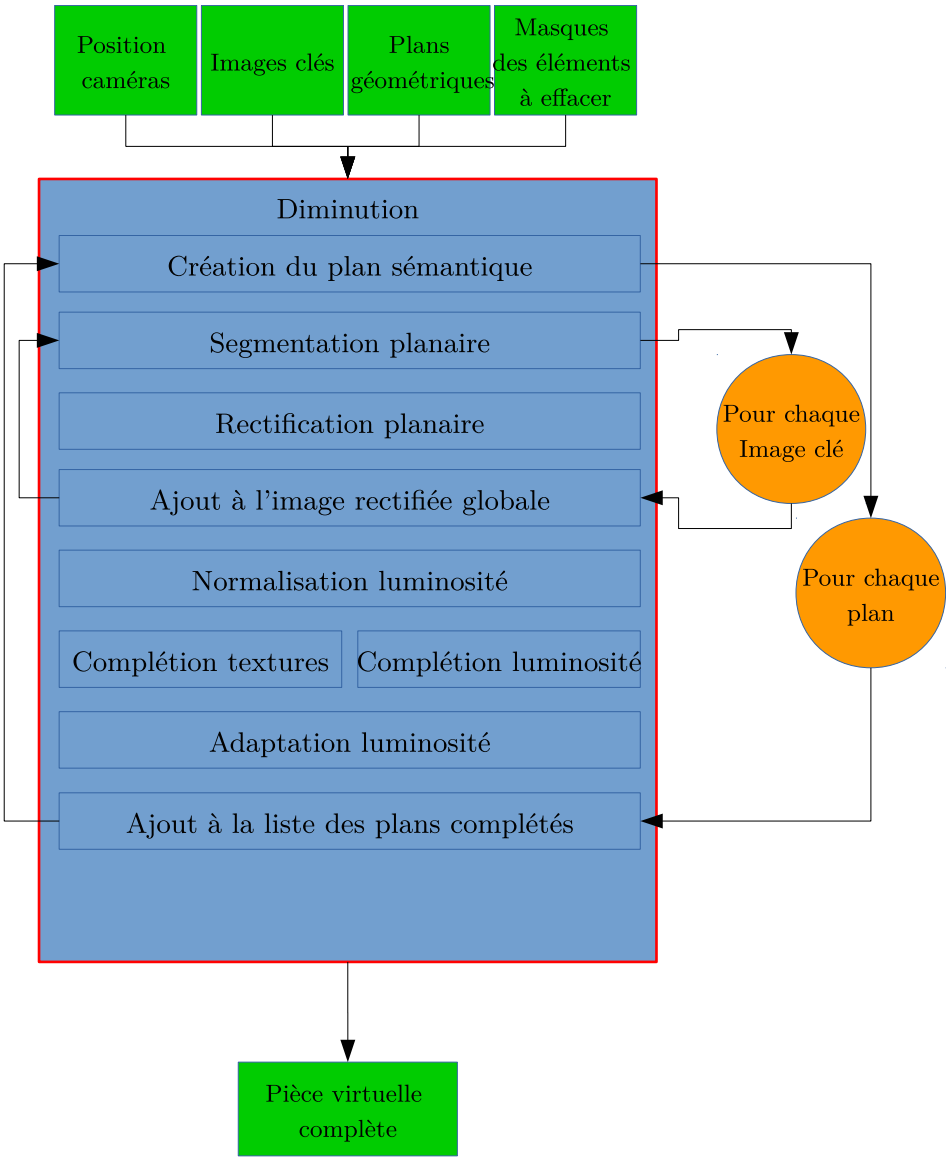


FIGURE 6.3 – Détails de la brique de diminution de *RemoveMyKitchen*

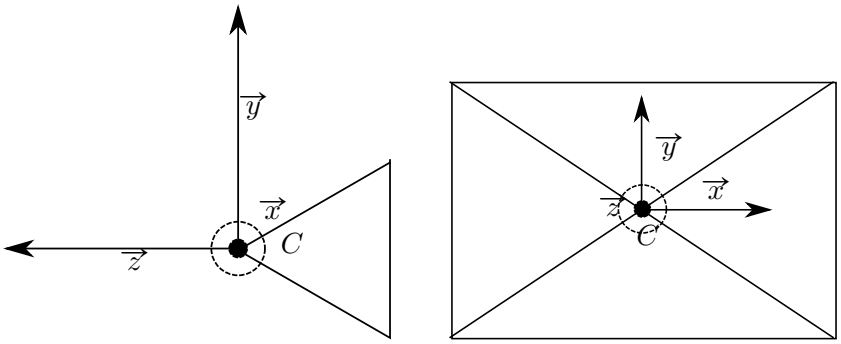


FIGURE 6.4 – Schéma de la convention utilisée pour le repère caméra.

plan Δ dans l'image I_i dans l'image rectifiée I_Δ. Pour cela, nous associons quatre points de I_i et quatre points de I_Δ.

Nous proposons de trouver des candidats à partir des coins du plan Δ. Les coins du plan Δ sont connus en 3D suite à l'étape de reconstruction de la pièce présentée dans la sous-section 6.2.1. Nous définissons plusieurs types de plans selon qu'ils soient visible ou pas par rapport à l'image que l'on veut rectifier. Nous adoptons, à partir de maintenant, la convention de la figure 6.4 pour le modèle caméra.

Définition 6.2.1. Soit un point $\mathbf{p} \in \mathbb{P}^4$, l'espace projectif de $\mathbb{R}^3$ et C une caméra définie par les matrices $\mathtt{K}$ de taille 3×4 (paramètres intrinsèques) et $\mathtt{T}^{-1}$ de taille 4×4 (contenant la rotation $\mathtt{R}$ et la translation $\mathbf{t}$ associées à la pose caméra). $\mathbf{p}$ est *observable*[2] selon C si :

$$(\mathtt{T}^{-1}\mathbf{p})^z < -f \tag{6.1}$$

où $(\mathtt{T}^{-1}\mathbf{p})^z$ est la coordonnée selon $\vec{z}$ de $\mathtt{T}^{-1}\mathbf{p}$ et f est la focal de la caméra C.

Définition 6.2.2. Soit Δ un plan fini délimité par les 4 coins $\mathbf{c}_i, i \in [1,4]$ et soit C une caméra définie par $\mathtt{K}$ et $\mathtt{T}$.
Δ est *observable* selon C s'il existe au moins un coin $\mathbf{c}_i$ qui est observable selon C.
Δ est *strictement observable* selon C si tous les coins $\mathbf{c}_i$ sont observables selon C.
Δ est *non observable* selon C si aucun des coins $\mathbf{c}_i$ n'est observable selon C.
Nous appelons *zone observable* d'un plan fini Δ (notée $v(\Delta)$) l'ensemble des points observables de Δ. On a donc :
Δ est observable selon C si $v(\Delta) \neq \varnothing$
Δ est strictement observable selon C si $v(\Delta) = \Delta$
Δ est non observable selon C si $v(\Delta) = \varnothing$
La figure 6.5 montre plusieurs exemples de plans définis ci-dessus.

Soit un Δ un plan observable, on note $\mathcal{L}_\Delta$ l'ensemble des coins de Δ réordonnés dans l'ordre croissant selon l'axe $\vec{z}$ de la caméra et orienté par la convention de la figure 6.4. Nous construisons à partir de $\mathcal{L}_\Delta$ un ensemble $\mathcal{L}_\Delta^v$ où tous les coins sont visibles. Soit $\mathbf{c}_v \in \mathcal{L}_\Delta$ le coin le « plus loin » de C :

$$\mathbf{c}_v = \text{argmax}_{\mathbf{c}_v} \left(\mathtt{T}^{-1}\mathbf{c}_v^z < \mathtt{T}^{-1}\mathbf{c}^z \right) \tag{6.2}$$

Nous définissons également une relation d'équivalence entre deux plans finis

2. Dans la littérature [HZ04], on dit plutôt qu'un point p valide le *cheirality test* . Le terme n'ayant pas d'équivalent français, nous utilisons le terme *observable*. Cependant, si le lecteur veut plus d'information à ce sujet, il devra plutôt utiliser le terme anglophone.

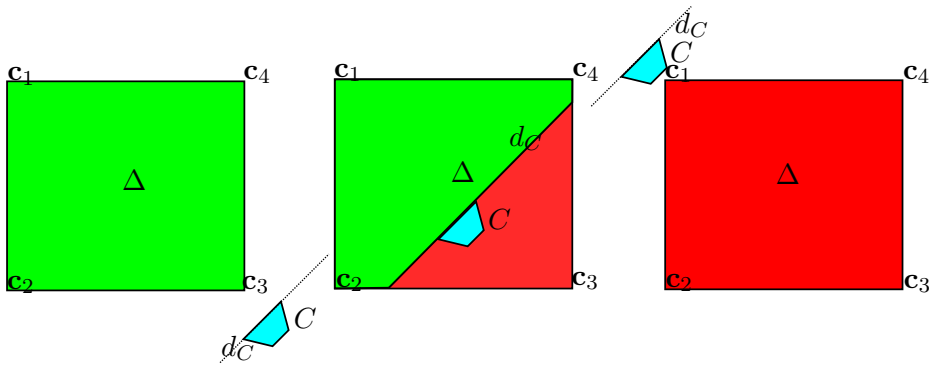


FIGURE 6.5 – Exemples de plans observables (milieu), strictement observables (gauche) et non observables (droite) ar la caméra C. Chaque plan est représenté vu de dessus. La zone observable (respectivement non observable) est coloriée en vert (respectivement en rouge). La droite d_C représente l'intersection entre le plan géométrique généré par le plan-image et le plan Δ (encadré par les coins $\mathbf{c}_i$).

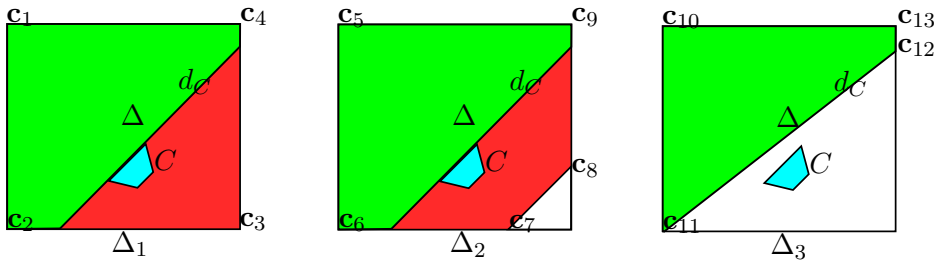


FIGURE 6.6 – Exemple de relations d'équivalence entre plusieurs plans finis représentant un même plan géométrique. Dans cet exemple, nous avons Δ_1 (composé des coins $\{\mathbf{c}_i, i \in [|1,4|]\}$ et Δ_2 (composé des coins $\{\mathbf{c}_i, i \in [|5,9|]\}$) qui sont équivalents. Mais Δ_1 et Δ_3 (composé des coins $\{\mathbf{c}_i, i \in [|10,13|]\}$) ne partagent pas la même zone visible ; ils ne sont donc pas équivalents.

Définition 6.2.3. Soient deux plans finis Δ_1 et Δ_2 d'une même plan géométrique. Soit C une caméra On dit qu'ils sont équivalents selon C si

$$v(\Delta_1) = v(\Delta_2) \tag{6.3}$$

La figure 6.6 montre quelques exemples de plans équivalents.
Plusieurs cas se dessinent :

— Soit le plan est strictement observable : tous les coins ont une projection dans le plan image de C. Nous posons simplement $\mathcal{L}^v_\Delta = \mathcal{L}_\Delta$

— Soit le plan est non observable : $\mathcal{L}^v_\Delta = \varnothing$. Nous ne traitons pas ce plan pour cette prise de vue.

— Soit le plan est observable : Nous cherchons un plan strictement observable équivalent. Pour cela, nous notons $\mathcal{L}^{nv}_\Delta \subset \mathcal{L}_\Delta$ l'ensemble des points non visibles. Nous cherchons pour chaque coin $\mathbf{c} \in \mathcal{L}^{nv}_\Delta$ un ou plusieurs points observables. Pour cela, nous considérons la fonction de projection sur le plan image de C proj suivante :

$$\begin{aligned} \text{proj} : \mathbb{R}^3 \times \mathcal{L}^v_\Delta &\to \mathbb{R}^3 && (6.4)\\ (\mathbf{p}, \mathbf{p}_v) &\mapsto \mathbf{p} + \alpha\overrightarrow{\mathbf{p}_v - \mathbf{p}} && (6.5) \end{aligned}$$

où α est calculé de sorte que proj(p) soit dans le plan image de C, $\mathbf{p}_v \in \mathcal{L}^v_\Delta$ et $\mathbf{p} \in \mathcal{L}^{nv}_\Delta$.

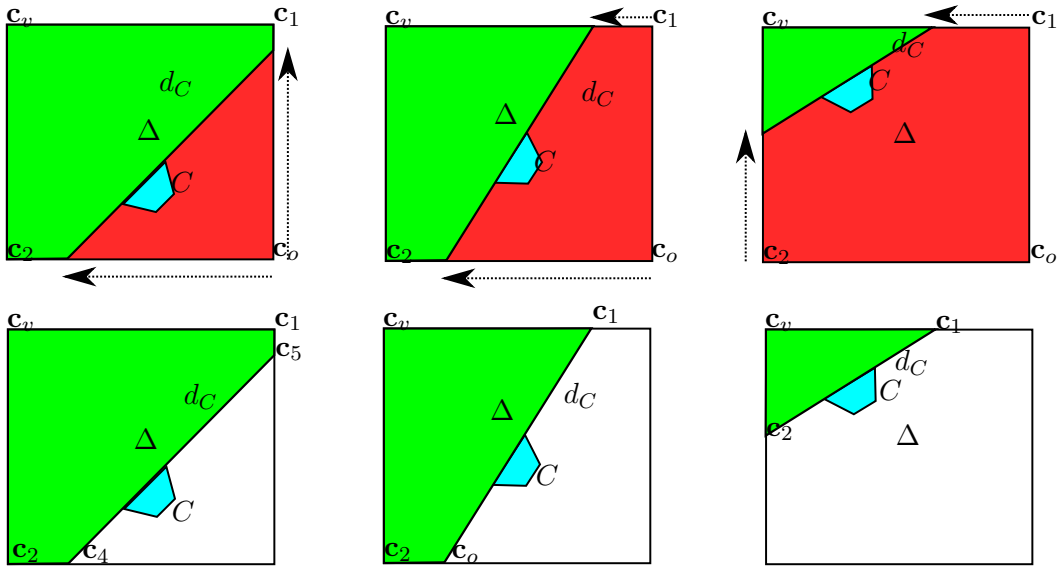


FIGURE 6.7 – Recherche d'un plan strictement observable équivalent à un plan observable selon une caméra C : présentation des trois cas possibles. Première ligne : les trois types de plans observables. Deuxième ligne : les équivalents strictement observables de chaque plan de la première ligne.

Trois cas se présentent comme montrés sur la figure 6.7.

— Seul un coin est non observable, nous appliquons proj à ce dernier avec ces deux coins voisins (et donc observables).
— Deux coins sont non observables, nous appliquons proj à chaque coin non observable avec pour chacun son coin voisin observable.
— Un seul coin est observable, nous appliquons proj à ses coins voisins non observables.

Nous retrouvons donc avec un plan strictement observable.

Nous projetons chaque coin dans le plan image. Ensuite, nous calculons l'intersection entre le quadrilatère projeté et formé des coins du plans et du domaine de l'image. Cela permet d'avoir la zone à rectifier ainsi que des points dont nous pouvons connaître la position dans l'image rectifiée pour calculer l'homographie `H` associée.

Ajout dans l'image rectifiée globale

Une fois l'homographie `H` calculée, nous l'appliquons à $I_{\Delta,i}$ la partie de l'image de la frame considérée qui appartient au plan Δ. Cependant, nous ne faisons pas un ajout brut du projeté de $I_{\Delta,i}$ en raison de deux éléments :

— Il peut y avoir une donnée déjà existante dans l'image rectifiée. Il faut donc décider quelle donnée garder.
— La frame a une exposition différente. Il faut donc réadapter l'exposition de la frame traitée.

6.2.2.1 Choix de la donnée

Définition 6.2.4. Soit I une image définie sur un domaine $O \subset \mathbb{N}^2$ et à valeur dans $\mathbb{R}^n$. Nous appelons le *domaine connu* $D(I) \subset O$ l'ensemble des pixels de O qui possèdent une valeur de couleur autre que le noir.

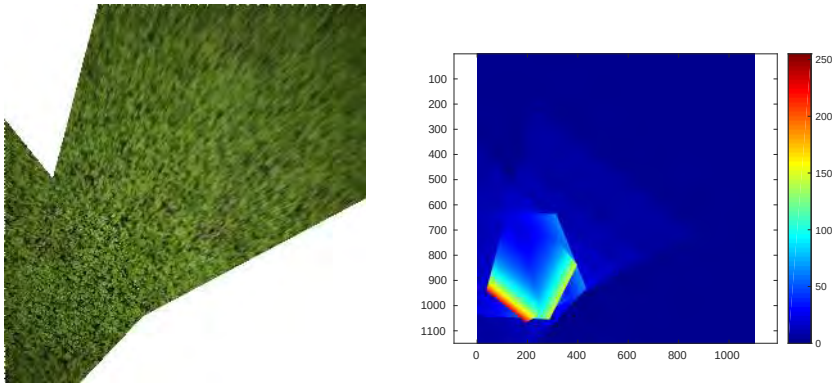


FIGURE 6.8 – Reconstruction d'une image rectifiée I_Δ à partir de plusieurs points de vue (image de droite). La sélection est assurée par la carte de confiance (image de gauche).

Notons $I_{\Delta,i}$ l'image contenant la rectification de la capture i par rapport au plan Δ. Cette image a des dimensions égales à I_Δ. Si $D(I_{\Delta,i}) \cap D(I_\Delta) \neq \varnothing$, nous déterminons quelle donnée doit être gardée dans I_Δ. Pour cela, nous utilisons le critère de confiance défini durant le chapitre 4.

Pour rappel, grâce à ce critère, nous disposons pour un plan Δ et une caméra C une carte de confiance $\mathcal{C}$ (*cf.* équation 7.1 du chapitre 4) :

$$\mathcal{C} : \begin{cases} U \times V & \longrightarrow \mathbb{R} \\ p & \longmapsto \mathcal{C}(p) = \text{trust}(\mathbf{P}) \end{cases} \tag{6.6}$$

où trust est la fonction de confiance construite à partir de ce critère radiométrique.

Nous changeons ainsi les pixels connus de I_Δ si la carte de confiance de $I_{\Delta,i}$ en ces pixels est supérieure à celle de I_Δ soit pour tout pixel p

$$I_\Delta(p) \leftarrow \mathbb{1}\left(\mathcal{C}_i(p) \geqslant \mathcal{C}(p)\right) I_{\Delta,i}(p) + \mathbb{1}\left(\mathcal{C}_i(p) \leqslant \mathcal{C}(p)\right) I_\Delta(p) \tag{6.7}$$

Cela permet de sélectionner le pixel de meilleure résolution.

Le domaine connu $D(I_{\Delta,i})$ est également actualisé pour tenir compte des données utilisées :

$$D(I_{\Delta,i}) \leftarrow \mathbb{1}\left(\mathcal{C}_i(p) \geqslant \mathcal{C}(p)\right) D(I_{\Delta,i}) \tag{6.8}$$

La figure 6.8 montre un exemple de I_Δ construite à partir de plusieurs points de vue.

Correction de l'exposition et normalisation de la luminosité

Comme expliqué dans le chapitre 5, nous cherchons pour chaque frame i son coefficient d'exposition α_i à partir des données projetées dans l'image I_Δ. Pour cela, nous résolvons le problème d'optimisation défini par l'équation 5.2 sur $\bigcap_i D(I_{\Delta,i})$. La valeur cible choisie est la valeur de α de la *frame* dont la valeur moyenne de sa carte de confiance est $\mathcal{C}_i$ la plus élevée. La figure 6.9 montre l'image I_Δ avant et après la correction de l'exposition. L'image rectifiée I_Δ est ensuite séparée entre sa carte de texture T_Δ et sa carte de luminosité $\mathrm{C}_{\mathrm{lum}\Delta}$. La figure 6.10 montre un exemple de cette étape.

Complétion de la texture

Nous disposons, à ce stade, pour chaque plan Δ d'une carte de texture T_Δ, d'une carte de luminosité $\mathrm{C}_{\mathrm{lum}\Delta}$ et d'un masque Ω_Δ des zones à compléter. Pour les cartes de texture et les cartes de luminosité, nous effectuons un processus de complétion différent.

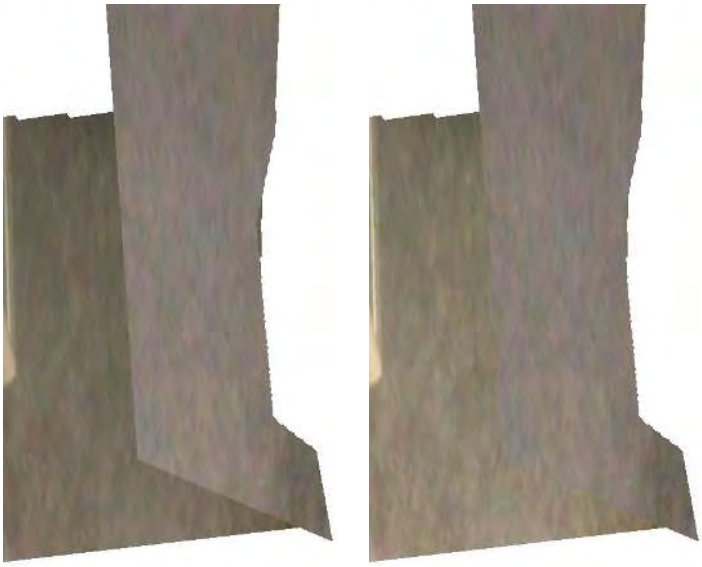


FIGURE 6.9 – Correction de l'exposition. Avant : sans correction. Après : avec correction.

Complétion de la carte de texture Pour rappel, la carte de texture T_Δ correspond au motif de la texture appliqué au plan Δ indépendamment de la luminosité appliquée à celui-ci. Pour le compléter, nous avons le choix entre effectuer un processus d'*inpainting* sur la zone masquée Ω_Δ ou d'effectuer une synthèse de texture à partir d'une tuile. Le deuxième choix se justifie dans le cas où nous n'avons pas assez d'information dans l'image rectifiée pour pouvoir faire efficacement de l'*inpainting*. L'*inpainting* complète un masque d'une image à partir des informations situées ailleurs dans l'image. Il faut donc suffisamment d'informations pour pouvoir compléter correctement le masque. À l'inverse, la synthèse de texture va créer une nouvelle image à partir d'une tuile génératrice. Elle a juste besoin de connaître la tuile considérée pour fonctionner ; cette tuile est d'ailleurs connue grâce à l'utilisateur dans la première étape. La figure 6.11 montre un exemple où l'*inpainting* n'arrive pas à remplir le masque de l'image rectifiée ; à l'inverse, le résultat est bien meilleur en faisant une synthèse de texture.

Le choix va se faire en fonction du type de structure et de la quantité d'information disponible.
Pour déterminer le type de la texture, nous utilisons les définitions et la méthode *a contrario* proposée au chapitre 3.
Pour quantifier la quantité d'information disponible, nous devons nous poser les questions suivantes. Comment évaluer la quantité d'information que nous avons à notre disposition ? Quel est le seuil pour déterminer si nous avons suffisamment d'information pour procéder à une complétion par une approche d'*inpainting* tout en ayant un résultat convenable. Premièrement nous utilisons l'expression suivante pour quantifier :

$$|D(T_\Delta)| = \frac{\sum_{\overrightarrow{o} \in \mathcal{O}} \text{occurence}(\overrightarrow{o})}{|U|} \tag{6.9}$$

où $\mathcal{O}$ est le numérateur est le nombre des *offsets* et $|U|$ est le nombre de pixels dans l'image. Nous estimons que nous avons suffisamment d'information pour appliquer une méthode d'*inpainting* si $|D(T_\Delta)|$ est supérieur à un seuil τ. L'idée de cette expression est de considérer que plus nous avons d'*offsets* à disposition par rapport aux dimensions de l'image, plus l'analyse statistique qui en découle sera pertinente. En fonction du type de structure de la texture et de la quantité d'information disponible, nous définissons

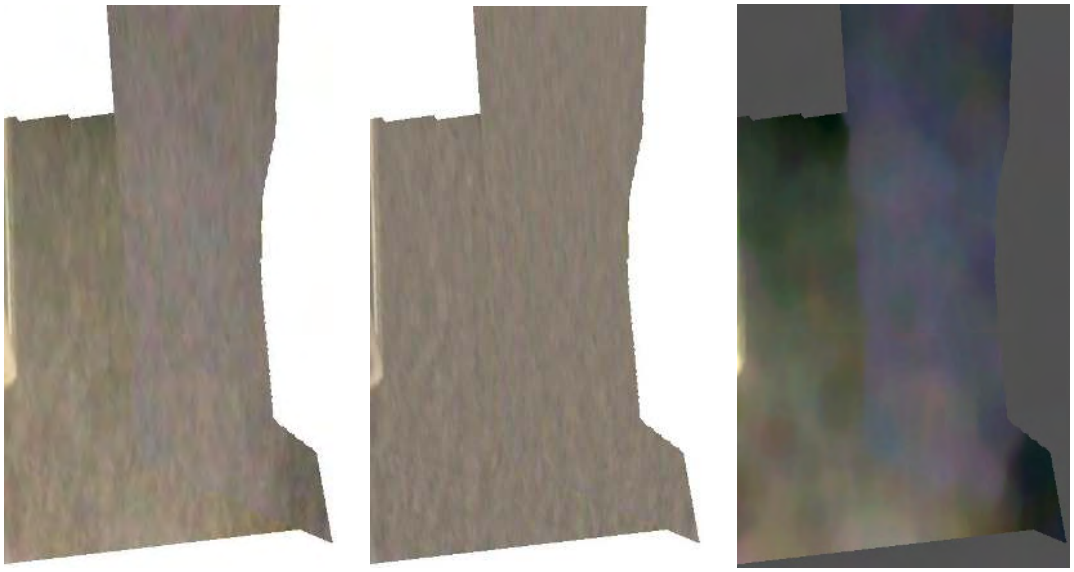


FIGURE 6.10 – Normalisation de la luminosité. L'image rectifiée I_Δ (gauche) est divisée entre sa carte de texture (milieu) et sa carte de luminosité (droite).

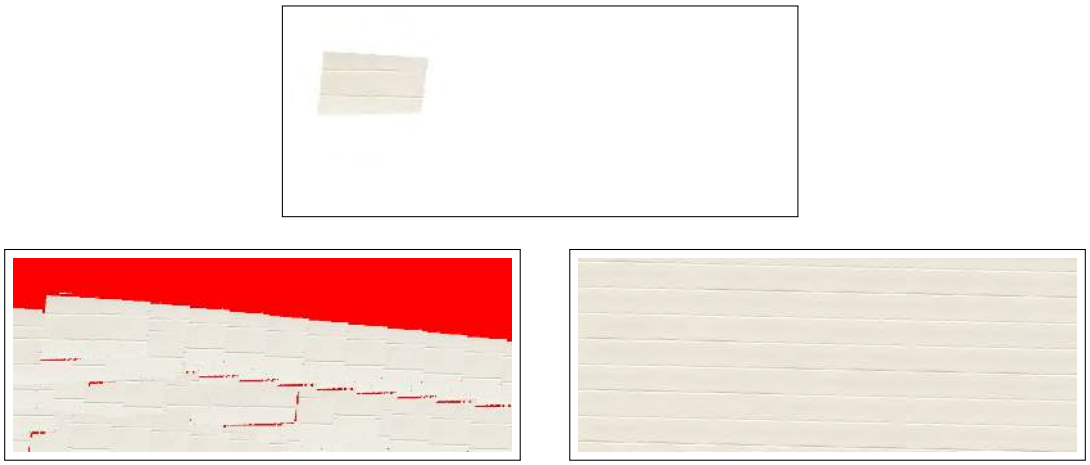


FIGURE 6.11 – Complétion d'une image rectifiée (première ligne). Sur la deuxième ligne, l'image est complétée via l'*inpainting* (image de gauche) ou elle est générée via une synthèse de texture (image de droite). La zone rouge signifie que nous n'avons pas pu trouver d'*offsets* pour copier l'information connue dans cette zone.

FIGURE 6.12 – Correction du décalage après la synthèse. À partir de l'image rectifiée (première ligne, gauche), une nouvelle image est créée par synthèse (première ligne, droite). Cette dernière est décalée (deuxième ligne, gauche) pour correspondre avec l'image incomplète en cherchant un vecteur de translation. Enfin, la partie vide due à la translation est complétée (deuxième ligne, droite) par une approche d'*inpainting* par *offsets*.

un seuil τ. Actuellement, τ est fixé empiriquement de sorte que 15% de pixels connus suffisent pour faire de l'*inpainting*, la manière pour déterminer τ n'étant pas immédiate. Une piste pour calculer dynamiquement ce seuil est abordé dans les perspectives. Ce seuil agit de la manière suivante :

- Si $|D(T_\Delta)| > \tau$, nous utilisons la méthode d'*inpainting* expliqué au chapitre 3 pour compléter la zone inconnue.
- Si $|D(T_\Delta)| < \tau$, nous appliquons la méthode d'*inpainting* basé *PatchMatch* [Bar+09] avec la diffusion comme initialisation si la texture est classée comme stochastique. Sinon, nous appliquons une synthèse de texture [Kwa+03] en utilisant une tuile contenant le motif répétitif.

Dans ce dernier cas, il est cependant nécessaire de recaler l'image synthétisée par rapport à la position d'origine de la tuile utilisée par la synthèse. La figure 6.12 illustre le procédé. Le vecteur de translation est trouvé en cherchant une correspondance entre l'image rectifiée incomplète et l'image synthétisée dans la zone connue $D(I_\Delta)$. La correspondance est effectuée par *PatchMatch*. Il nous suffit d'appliquer la translation et de compléter les zones laissées vide par la translation.

Finalement, nous obtenons une image I_Δ où toutes les parties vides ou masquées ont été complétées.

6.2.2.2 Adaptation de la luminosité

La carte de luminosité $C_{\text{lum}\Delta}$ contient plusieurs types d'informations qui ne peuvent être sur la même image. Nous avons à la fois la projection d'une source lumineuse sur la texture du plan et à la fois, si la surface n'est pas lambertienne, les tâches spéculaires dues à la réflexion des rayons lumineux. Comme expliqué durant le chapitre 5, nous divisons cette carte en deux images [STB18] : une image $C_{\text{lum}\Delta,s}$ qui contient l'information spéculaire et une image $C_{\text{lum}\Delta,d}$ qui contient l'information diffuse.
Pour effectuer la séparation, nous devons détecter les tâches spéculaires. Pour cela, nous utilisons la méthode de MORGAND et TAMAAZOUSTI [MT14] qui détecte les tâches

spéculaires dans une image.
Ensuite, nous utilisons l'approche de Said *et al.* [STB18] pour reconstruire les tâches spéculaires présentes dans $\mathrm{C}_{\mathrm{lum}\Delta,s}$.
Quant à $\mathrm{C}_{\mathrm{lum}\Delta,d}$, nous complétons d'abord les isocontours traversant le masque. Nous appliquons ensuite une méthode d'*inpainting* basée diffusion qui est guidée par les valeurs de ces isophotes.

L'image finale rectifiée est donc la suivante :

$$I_\Delta = T_\Delta + (\mathrm{C}_{\mathrm{lum}\Delta,d} + \mathrm{C}_{\mathrm{lum}\Delta,s}) \tag{6.10}$$

Reprojection

Une fois l'image rectifiée globale I_Δ calculée pour chaque plan Δ de la scène d'intérieur, nous projetons dans chaque prise de vue toutes les images rectifiées I^i_Δ par l'inverse de chaque homographie $\mathtt{H}_i^\Delta$ associée. Le critère de confiance du chapitre 4 assure par ailleurs qu'il n'y ait pas flou artificiel dans le rendu final et, par filtrage, que toutes les données soient ramenées à la résolution attendue.

6.2.3 Résultats

Conditions d'expérience

L'application est codée en plusieurs langages de programmation. La brique de diminution est codée en C++ et utilise des bibliothèques comme CImg [Tsc12] (traitement de l'image) et Ceres [A+16] (optimisation non linéaire). La brique est ensuite connectée à une application Android via une interface JNI. Les interactions utilisateurs sont codées directement au sein de l'environnement Android.
Nous avons testé sur plusieurs cuisines et pièces d'intérieur. Les données d'entrée sont constituées généralement de trois à quatre images. Le cas de l'ensemble constitué de trois images correspond à une situation où la pose caméra de la quatrième prise n'était pas optimale. Nous avons donc préféré la retirer du processus.
L'acquisition des données se fait sur une tablette du marché sous système Android. Celles-ci sont ensuite transférées sur un ordinateur qui exécutera le processus de génération de la géométrie de la pièce et de la diminution.
Pour trois/quatre images, le temps de calcul est d'environ deux minutes.

Observations et limites

Les figure 6.13 nous montre un premier ensemble de résultats dans le case d'une cuisine. Nous pouvons constater un bon rendu notamment au niveau de la luminosité. Notons également la présence d'éléments du sol sur l'image de droite de la deuxième ligne alors que cette information n'est pas présente sur l'image de gauche originale. Le résultat de la troisième ligne montre cependant un mauvais recalage du sol dû au manque d'information de ce plan dans les autres images. Un résultat analogue est montré dans le cas d'un *open-space* dans la figure 6.14. La figure 6.15 nous montre un exemple où la diminution s'est moins bien déroulée. Cela est dû à une mauvaise estimation de la géométrie de la pièce et une mauvaise optimisation des poses caméra. Cela est visible notamment au niveau de la projection d'une partie de la fenêtre de la troisième image sur la deuxième image. En terme de complétion de la texture des plans, les murs de la pièce montrée en figure 6.13 ont été complétés par l'*inpainting* vu au chapitre 3 tandis que le sol a été reconstitué via la synthèse d'un tuile du carrelage. On peut d'ailleurs constater le recalage de la texture synthétisée n'a pas été correctement effectuée. Dans

les autres figures (figure 6.14 et figure 6.15), tous les plans ont été complétées avec la méthode d'*inpainting* du chapitre 3. Notons que le mur de briques de la figure 6.14 a été mal reconstruit car la structure est difficilement identifiable.

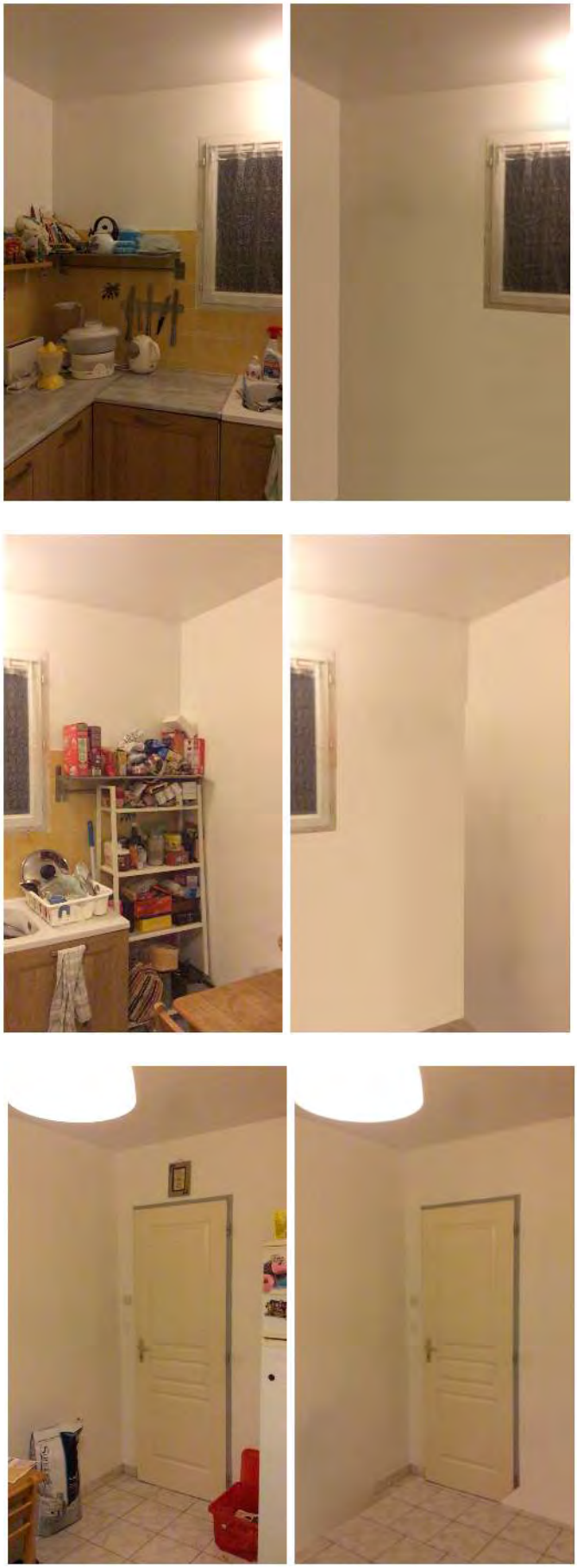


FIGURE 6.13 – Effacement des éléments d'une cuisine à partir de 3 points de vue avec *RemoveMyKitchen*. Notons dans la deuxième prise de vue (deuxième ligne) la projection d'une partie du sol alors qu'on n'a pas d'information dans cette prise de vue. En effet, le sol est reconstruit à partir de la troisième prise de vue (troisième ligne).

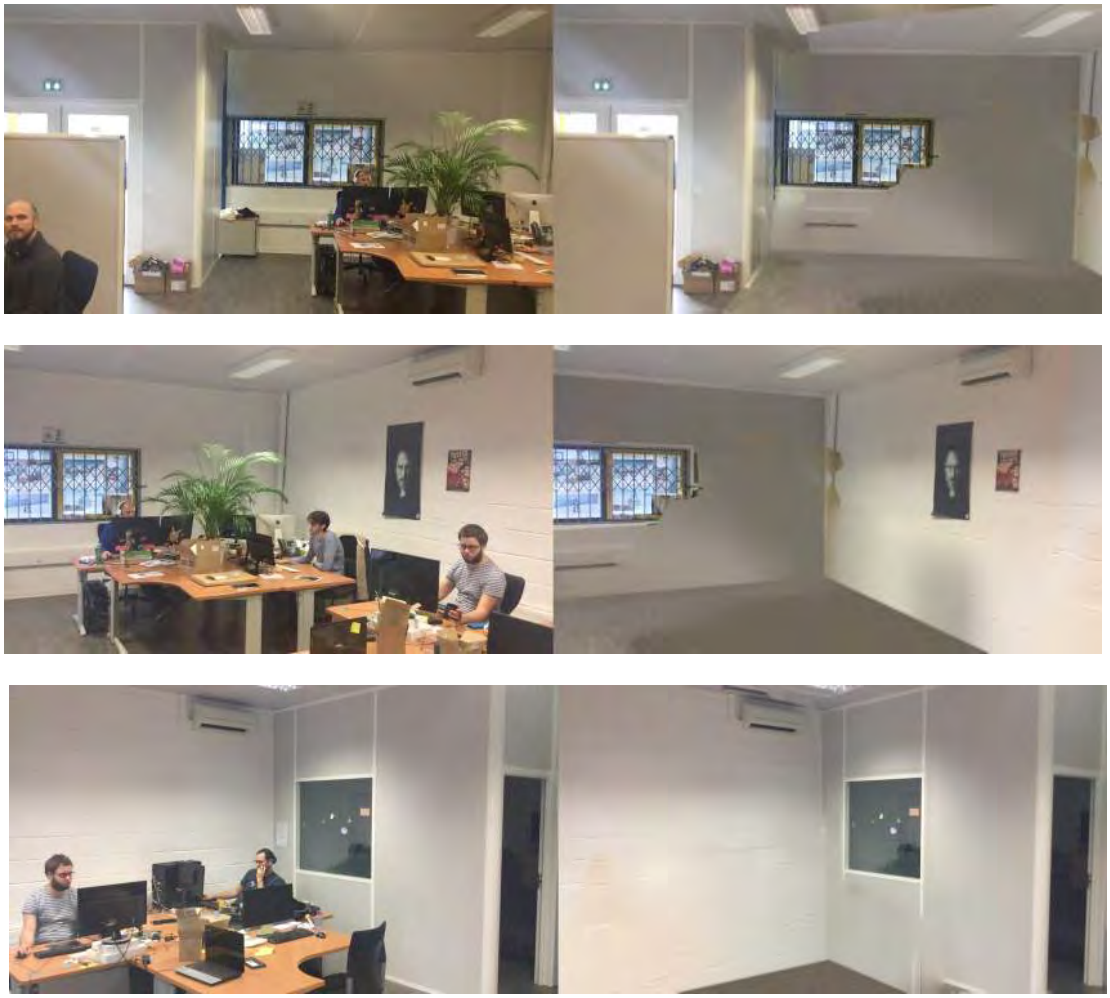


FIGURE 6.14 – Effacement des éléments d'un *open-space* avec *RemoveMyKitchen*. Si la géométrie est bien respectée dans les premières prises de vue (première et deuxième ligne), celle-ci a été moins bien reconstruit dans la zone visible par la troisième prise de vue (troisième ligne) vu que l'intersection attendue entre le plan du sol et le mur de droite n'est pas la même que l'intersection obtenue en calculant la géométrie de la pièce.

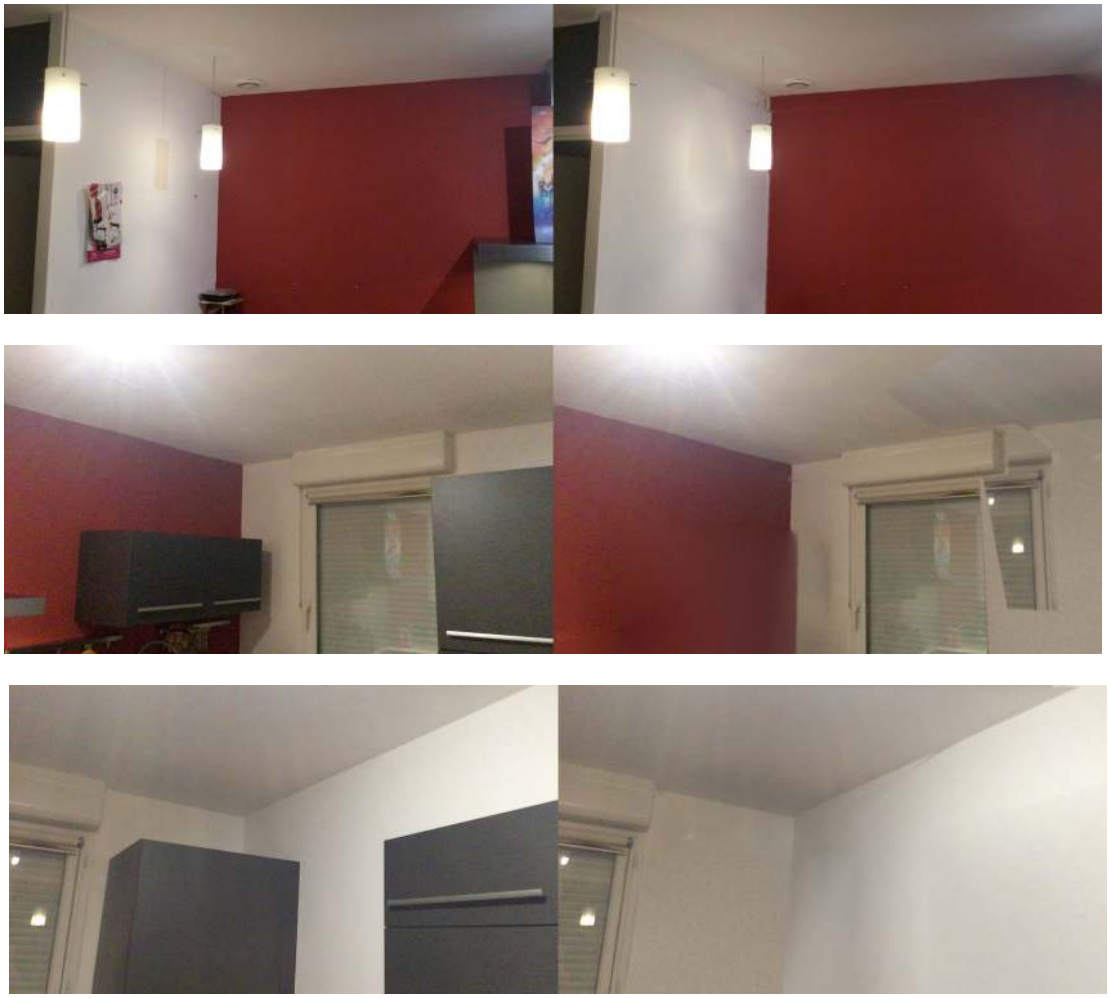


FIGURE 6.15 – Effacement des éléments d'une autre cuisine avec *RemoveMyKitchen*. Nous observons une propagation réaliste de la luminosité dans les zones masquées. Cependant, il faut noter aussi la mauvaise projection de la fenêtre de la troisième prise de vue (troisième ligne) dans la deuxième prise de vue (deuxième ligne).

6.3 Conclusion et perspectives

Nous avons abordé dans ce chapitre les applications possibles de nos contributions dans un cadre de géométrie planaire. Nous avons, pour cela, présenté deux applications développées au sein de l'équipe R&D d'Innersense dans lesquelles nous avons intégré les contributions. Ces deux applications étaient des preuves de concept mais que servent de base pour l'application détaillée dans le chapitre 8. La première application proposait de reconstruire une géométrie simple à partir d'une seule image. Malgré cette limitation, nous avons montré qu'il était possible de mettre en place un scénario de Réalité Diminuée. La deuxième application est une extension de la première. Elle permet, à partir de quelques prises de vue, de produire un modèle géométrique complet de la pièce. Il est alors possible d'effectuer une diminution pour enlever tous les éléments de la pièce. On peut alors fournir un rendu réaliste de la pièce vidé où l'utilisateur peut, encore de manière statique, ajouter des meubles virtuels.
Nous abordons maintenant les perspectives envisagées pour améliorer les résultats proposés.

Amélioration de la reconstruction

Comme vu ci-dessus, une mauvaise estimation de la pièce et/ou des caméras peut réduire la qualité du processus de diminution et aboutir à des résultats de rendu non idéaux : le non alignement des frontières entre 2 plans est particulièrement visible, notamment entre 2 plans de couleurs différentes, comme à la figure 6.15. Cela est dû notamment au faible nombre d'images utilisées pour effectuer cette reconstruction. De plus, les plans de la pièce peuvent être faiblement texturés. On peut donc manquer d'information de texture dans les images ce qui conduit à une dérive de l'estimation des poses caméras. Une extension est proposée par Gohard qui s'appuie sur la détection de segment, le lecteur intéressé peut aller voir sa thèse

Choix dynamique du seuil de décision τ

Comme dit dans la sous-section précédente, le seuil τ, en-dessous duquel on considère que nous n'avons pas suffisamment d'information pour effectuer l'*inpainting*, est assez naïf car fixé selon les dimensions de l'image rectifiée I_Δ. Une idée possible est d'analyser la représentation des premiers *offsets* les plus influents dans une texture stochastique en fonction des dimensions de l'image. Si les dimensions de l'image diminuent, la proportion des premiers *offsets* les plus influents dans l'ensemble des *offsets* va augmenter. L'approche *a contrario* définie dans le chapitre 3 sera biaisée car même les textures stochastiques seront considérées comme ayant un motif régulier. La valeur de τ peut être basée sur les dimensions à partir desquelles on ne peut plus distinguer le type de texture par l'approche *a contrario*. Une perspective peut-être d'appliquer ici un un test de significativité *significant test*.

Dans le prochain chapitre, nous appliquons nos contributions dans le cas d'une géométrie non-planaire.

Chapitre 7

Extension à la géométrie non planaire

Sommaire

Dans ce chapitre, nous faisons une extension au contexte considéré dans cette thèse, c'est à dire, un scénario de Réalité Diminuée de scènes d'intérieur. En effet, les pièces d'intérieur sont souvent constitués par des murs qui sont planaires. Nous proposons d'étudier dans ce chapitre la complétion de la texture d'un élément de surface non planaire. Le contexte de ce chapitre est celui de la reconstruction d'un objet texturé. Nour prenons, pour cela, le contexte de la complétion de textures d'un objet modélisable par des surfaces canal et dont le modèle est construit à partir d'un ensemble restreint de prises de vue.

7.1 Introduction

Au cours de la dernière décennie, de nombreuses méthodes et approches ont été proposées pour générer un modèle 3D d'un objet à partir d'un ensemble d'images. La

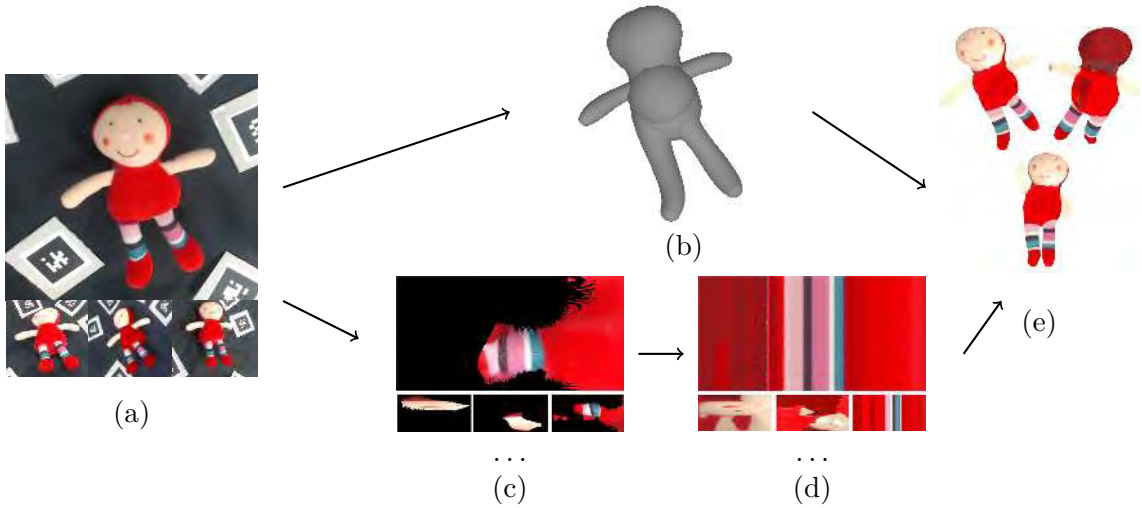
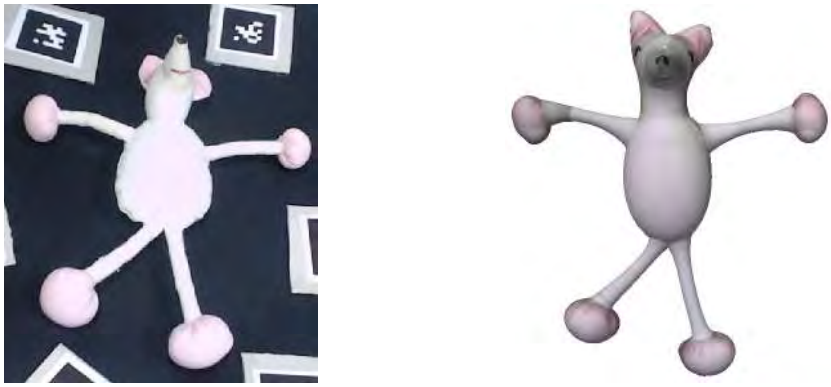


FIGURE 7.1 – Illustration du *pipeline* complet. (a) Acquisitions calibrées de l'objet à reconstruire. (b) L'objet est reconstruit avec son squelette. (c) Les textures apparentes sont extraites des images pour chaque branche du modèle. (d) Comme ces textures sont partielles (par exemple, nous ne pouvons pas voir le dos de la peluche ici), elles sont complétées par *inpainting*. (e) Les textures complétées sont appliquées à l'objet, qui peut être facilement animé.

plupart des approches sont basées sur le *Structure-from-Motion* et le *Multi-View Stereo* (MVS) [FH15], qui permet la reconstruction de l'objet à partir d'un ensemble non ordonné d'images [SSS06]. Ces méthodes fonctionnent bien si l'objet est suffisamment texturé, ce qui permet de trouver des ancres (points d'intérêt) pour créer des correspondances entre les images. Le modèle géométrique généré par ces méthodes classiques de reconstruction est généralement un nuage de points 3D, qui est ensuite triangulé pour générer un maillage triangulaire. L'étape finale consiste à appliquer la texture sur la géométrie reconstruite et pour cela (texturation), vise à fournir une texture cohérente pour le maillage à partir des images sources multiples, et, en particulier, à assurer une texture cohérente à travers les triangles voisins du maillage [WMG14]. Notons que, pour obtenir un modèle de bonne qualité, l'objet doit être visible dans suffisamment d'images prises sous différents points de vue.

Dans ce travail, nous traitons de la reconstruction d'une famille spécifique d'objets qui peuvent être représentés par un ensemble de surfaces canal (*branches*) [HC32]. En particulier, nous nous appuyons sur la méthode de reconstruction géométrique d'objets tubulaires récemment proposée par DURIX *et al.* [Dur+15 ; Dur+16] (voir figure 7.1) : à partir d'un nombre limité (généralement de deux à cinq) d'images calibrées (sous-figure 7.1a) ils génèrent un modèle géométrique de l'objet qui est composé par un ensemble de surfaces canal paramétriques (sous-figure 7.1b), *i.e.* un modèle par morceaux de surfaces canal. Un des avantages de cette méthode de reconstruction est qu'une reconstruction complète de l'objet peut être obtenue à partir d'un nombre très limité d'images, sans nécessairement avoir l'intégralité de la surface de l'objet capté. De plus, la reconstruction géométrique ne nécessite pas d'images de bonne qualité, ni d'un calibrage élaborée, et elle est capable de reconstruire des objets même s'ils ont une texture uniforme –sur laquelle les points d'intérêt seraient difficile à obtenir (*cf.* figure 7.1). Nous proposons d'étendre et de compléter leur chaîne de traitement en **texturant le modèle géomé-**



(a) (b)

FIGURE 7.2 – Une peluche avec une texture très uniforme est reconstruite et texturée. Ici, seule la texture avant (a) est visible sur les images, et la texture arrière (b) est estimée par la méthode proposée. On note que malgré l'absence de points d'intérêt sur le modèle initial, l'objet entier est reconstruit.

trique reconstruit.

Le problème majeur à résoudre est la complétion de la texture pour les parties de l'objet qui ne sont pas visibles sur les images d'entrée ou qui sont cachées à cause des occlusions. Nous proposons une méthode de texturation qui applique les images d'entrée sur la surface paramétrique de l'objet. Les parties de la surface qui ne sont visibles dans aucune image d'entrée sont complétées par un processus d'*inpainting*. Nous proposons une nouvelle méthode similaire à la technique classique de texturation qui sélectionne, pour chaque surface canal 3D, les images les plus appropriées et les fusionne pour appliquer la texture complétée à la surface. Le processus est régulé par un critère de confiance qui sélectionne les images en fonction de leur position et de leur orientation par rapport à la surface (sous-figure 7.1c). Nous proposons ensuite deux méthodes pour compléter la texture dans les parties cachées de la surface en fonction du type de texture (sous-figure 7.1d). Notre méthode est basée sur un processus d'optimisation globale qui fusionne les images en tenant compte de leur différence d'exposition et en corrigeant les problèmes d'alignement. Une fois intégrée dans le pipeline d'origine, un modèle 3D texturé peut être généré à partir de quelques images d'entrée, même si l'intégralité de l'objet n'a pas été captée (sous-figure 7.1e).

L'avantage de l'approche proposée est la génération d'un modèle 3D texturé complet à partir de quelques images d'entrée qui peuvent ne proposer qu'un rendu partiel de l'objet, certaines parties ne sont visibles sur aucune image. Les principales contributions de ce travail sont (*i*) la reconstruction d'une texture pour chaque branche du modèle à partir des images d'entrée en choisissant l'image la plus appropriée, (*ii*) la fusion de la texture issue de différentes images avec correction d'exposition et d'alignement, et (*iii*) deux méthodes pour compléter la texture des parties des branches qui ne sont visibles par aucune caméra (voir figure 7.2b).

Le chapitre est organisé comme suit. La section 7.2 présente l'état de l'art de la texturation des modèles 3D reconstruits ; la section 7.3 présente les principales étapes du pipeline proposé et la section 7.4 décrit la méthode proposée pour ajuster l'exposition de la texture et compléter les parties manquantes. La section 7.5 présente quelques résultats et une discussion des limites, tandis que la section 7.6 conclut le chapitre avec les orientations futures et les améliorations, possibles ou à envisager, de la méthode proposée.

7.2 État de l'art

Dans cette section, nous passons en revue l'état de l'art de la texturisation des modèles 3D générés par différentes approches de reconstruction 3D.

7.2.1 Texturation d'un modèle 3D généré par MVS

Comme mentionné dans la section 7.1, l'approche de reconstruction 3D qui donne des résultats plus prometteurs est la stéréoscopie multi-vues (MVS) [FH15]. A partir d'un ensemble d'images couvrant toutes les parties visibles de l'objet, des correspondances entre les images peuvent être calculées et triangulées pour générer un nuage dense de points 3D [SSS06]. Il en résulte généralement un modèle 3D de l'objet capable de capter les plus petits détails avec un bon niveau de précision [Kna+17]. Une fois le modèle de maillage 3D généré, la dernière étape du processus est la texturisation du maillage, c'est-à-dire l'affectation d'une couleur ou d'une texture à chaque face du maillage. Souvent, comme étape de prétraitement, le maillage est simplifié avant la texturisation afin d'obtenir des triangles plus grands : alors que la géométrie globale peut être maintenue avec une précision suffisante, la texture est appliquée sur le modèle basse résolution, c'est à dire sur les grands triangles, afin de rendre le processus de texturisation plus efficace [Lue+02]. Le principal problème à résoudre lors de l'attribution d'une texture à une face est la sélection de l'image source la plus appropriée. Ainsi, la texturisation peut être considérée comme le problème de la sélection de la ou les meilleures vues pour chaque triangle, en tenant compte de différents paramètres, tels que la distance de l'image par rapport à la face, l'angle sous lequel la face est vue par l'image et, plus généralement, la qualité de l'image (effets de flou, éclairage *etc.*). De plus, pour obtenir un modèle 3D photoréaliste, la carte de texture appliquée au modèle doit être indépendante des conditions de lumière dans lesquelles les images originales ont été prises. La carte de texture doit être rendue correctement lorsque le modèle est montré avec un éclairage différent. Ceci nécessite de normaliser les images originales et de les mettre en correspondance pour en déduire un texture indépendante de l'éclairage, par exemple en optimisant leur cohérence de couleur [BMR01] ou en maximisant l'information commune entre les images projetées [Cor+13].

Les méthodes de texturisation peuvent être divisées en deux approches principales. Les *approches mono-vue* sélectionnent, pour chaque triangle indépendamment, la meilleure vue [GDA13]. Cette solution n'est optimale que localement, car elle génère inévitablement des artefacts visibles et des discontinuités entre les faces voisines ayant différentes images associées avec une exposition ou un éclairage éventuellement différents. Le mélange des images aux bords des faces peut atténuer le problème, sinon des techniques plus avancées utilisent l'étiquetage [Sin+08] ou la minimisation de l'énergie qui pénalise les discontinuités [WMG14]. Waechter *et al.* [WMG14] proposent une procédure d'étiquetage qui minimise une fonction d'énergie avec deux termes. L'un des termes de la fonction d'énergie prend en compte, non seulement l'angle de vue par rapport à la normale de la face, mais aussi la distance ainsi que le " flou " de l'image. Un terme de lissage mesure la différence entre les coutures droite et gauche.

Les *approches Multi-vues*, quant à elles, fusionnent un sous-ensemble d'images sources afin d'obtenir une image uniforme et cohérente. Cette approche globale peut souffrir d'une perte de qualité et de détails lors du mélange d'images de qualité ou résolution différentes, si elles sont prises à des distances différentes par exemple. Cela nécessite d'adopter un mélange pondéré qui favorise les images les plus proches du modèle [Cal+08]. Un autre problème est lié à l'estimation imparfaite de la géométrie et de l'ali-

gnement des caméras (*e.g.* étalonnage des caméras), ce qui, encore une fois, peut générer des artefacts dans l'image fusionnée. Pour surmonter ce problème, Bi *et al.* [BKR17] a proposé une synthèse par *patch* dans laquelle une vue synthétique est générée à partir de deux ou plusieurs images, en tenant compte des décalages géométriques tout en préservant la cohérence photométrique.

7.2.2 Texturation de modèles 3D générés par d'autres méthodes de reconstruction

D'autres méthodes de reconstruction utilisent la silhouette de l'objet [HS04], mais elles nécessitent un étalonnage précis (nécessitant la plupart du temps un environnement de capture dédié) et un plus grand nombre d'images (au moins 20). Le modèle généré, appelé le *visual hull*, est linéaire par morceaux et ne bénéficie d'aucune paramétrisation naturelle. La texturation à partir d'une image de référence est donc difficile, car aucune correspondance de la surface à l'intérieur de bords définissant la silhouette avec la texture rendue depuis un point de vue n'est donnée.

D'autre part, les objets créés par l'homme sont souvent de forme complexe mais peuvent être décomposées en formes plus simples, en particulier et en surfaces symétriques selon un axe de rotation, appelées surface tubulaires. Ces dernières peuvent être modélisées et reconstruites plus facilement avec des surfaces paramétriques.

Notre approche est similaire à celle de Chen *et al.* [Che+13], qui génère un modèle 3D à partir d'une seule image de l'objet avec l'interaction de l'utilisateur. Ils segmentent la forme complexe de l'objet en pièces plus petites et plus simples guidées par une série de gestes de « balayage » : ceux-ci permettent à l'utilisateur de définir deux dimensions d'un profil 2D et de « balayer » le long de l'axe courbe de l'objet. Ils récupèrent la texture de l'image par rétroprojection. Pour les parties cachées, deux approches sont proposées. En supposant que l'objet est symétrique, la texture visible est reflétée et appliquée sur les parties cachées OU non visibles. Cependant, des zones sans texture ne peuvent pas être complétées ainsi lorsque des points symétriques par rapport à l'axe de l'objet ne sont visibles, ni l'un, ni l'autre. Dans ce cas, l'*inpainting* est utilisé pour compléter la texture [Xia+11].

Plutôt que de s'appuyer sur une géométrie définie par un ensemble de points 3D, nous nous appuyons sur la reconstruction de la géométrie et de la topologie de l'objet à partir de squelettes [Dur+15 ; Dur+16]. Les modèles de texturation obtenus à l'aide d'une reconstruction paramétrique font face à d'autres défis. Comme le modèle est une approximation de la géométrie de l'objet réel, les problèmes d'alignements entre la caméra la géométrie peuvent être plus importants que ceux des pipelines classiques. Ils doivent être pris en compte lors du mélange et de l'enregistrement des images. Alors que les méthodes MVS peuvent uniquement reconstruire que ce qui est visible par au moins deux caméras, la reconstruction par squelette peut reconstruire les parties occultées de l'objet pour lesquelles les informations photométriques ne sont donc pas disponibles. Dans ce cas, la texturation doit remplir les "trous" de ces parties. Les méthodes d'*inpainting* détaillée dans le chapitre 2 peuvent être utilisé pour "halluciner" les régions sans texture en propageant la texture des régions voisines.

7.3 Paramétrisation et texturation de la surface

Dans les sections suivantes, nous présentons notre pipeline pour reconstruire la texture à appliquer au modèle 3D reconstruit à partir d'un ensemble d'images calibrées $(I_i)_{i=1\ldots n}$.

7.3.1 Reconstruction de la géométrie

Nous partons d'une reconstruction basée sur des squelettes, qui génère un ensemble de surfaces canal représentant l'objet 3D capturé [Dur+16 ; CG06]. Nous nous basons plus particulièrement sur l'approche de Durix *et al.* [Dur+16]. Cette reconstruction est basée sur une estimation de la projection du squelette de l'objet 3D sur chaque image ;

Tout d'abord, un objet est capté sur plusieurs images dont la prise de vue est calibrée. L'objet est segmenté sur chaque image avec l'algorithme semi-interactif GrabCut [RKB04]. Cette segmentation génère pour un chaque image un masque binaire de la projection de l'objet. À partir de cette image binaire et du calibrage, les projections du squelette sont calculées sur chaque image, et l'utilisateur associe les extrémités de chaque squelette. Cela permet d'établir une correspondance entre les topologies des différents squelettes (un par image) dont dérive une correspondance entre les projections d'une branche sur les différentes images. A l'aide de ces correspondances, chaque branche est triangulée en 3D et ainsi le squelette 3D est estimé. Chaque branche est reconstruite séparément, comme une surface de canal. Le squelette 3D ainsi généré est comme une union de surfaces paramétriques, des surfaces canal [HC32], qui approximent la forme complexe de l'objet réel. La surface canal est l'enveloppe d'une famille de sphères dont les centres et les rayons varient selon une courbe continue. Intuitivement, les objets 3D reconstruits ont un axe médian courbe. Les surfaces canal ont une paramétrisation naturelle de leurs surfaces [PP97], relativement régulière (sauf à la fermeture de la surface), ce qui permet d'appliquer facilement les textures que nous allons estimer OU reconstruire.

7.3.2 Un domaine paramétré pour la texture de la branche

Pour chaque branche b du modèle, nous visons à créer une image de référence I_b. Chaque branche du modèle est représentée par une surface canal paramétrique $S(t, \theta)$, où t se déplace le long de la courbe du squelette C, et θ tourne autour du point squelette $C(t)$ sur la surface. Si t varie dans un intervalle A de sorte que $\{C(t), t \in A\}$ décrive la courbe entière du squelette, nous pouvons prendre dans ce cas $A = [0, 1]$. Ainsi, $A \times [0, 2\pi]$ est le domaine paramétrique de la surface du canal.

Notre but est de reconstruire une image complète I_b sur le domaine $A \times [0, 2\pi]$ [1]. Ensuite, pour chaque image I_i utilisée pour la reconstruction géométrique de la branche, on peut projeter (en supposant que la caméra soit calibrée) la surface de la branche sur l'image (voir figure 7.3). Chaque pixel de I_i couvert par la projection d'une branche correspond à au moins un point 3D $S(t, \theta)$. Notons que, comme les branches sont générées par triangulation d'un squelette 2D à partir de différentes images, la rétroprojection d'une branche ne correspond pas exactement à l'image. Cependant, comme le masque de l'objet sur chaque image d'entrée est connu à partir de la reconstruction basée sur le squelette, nous n'utilisons que les pixels à l'intérieur du masque pour estimer la texture de chaque branche. Ce filtrage de masque évite de prendre en compte les pixels d'arrière-plan.

7.3.3 Gestion des occultations

Comme le modèle est composé de différentes surfaces canal, l'un des défis est d'identifier correctement la texture appartenant à une branche b, par exemple en cas d'(auto-)occultations entre les différentes parties de l'objet. Pour cela, nous nous appuyons sur un tampon de profondeur (*z-buffer*). Nous projetons tous les points de la surface sur

1. Notez que dans l'implémentation réelle nous considérons un domaine échantillonné et discret pour $A \times [0, 2\pi]$.

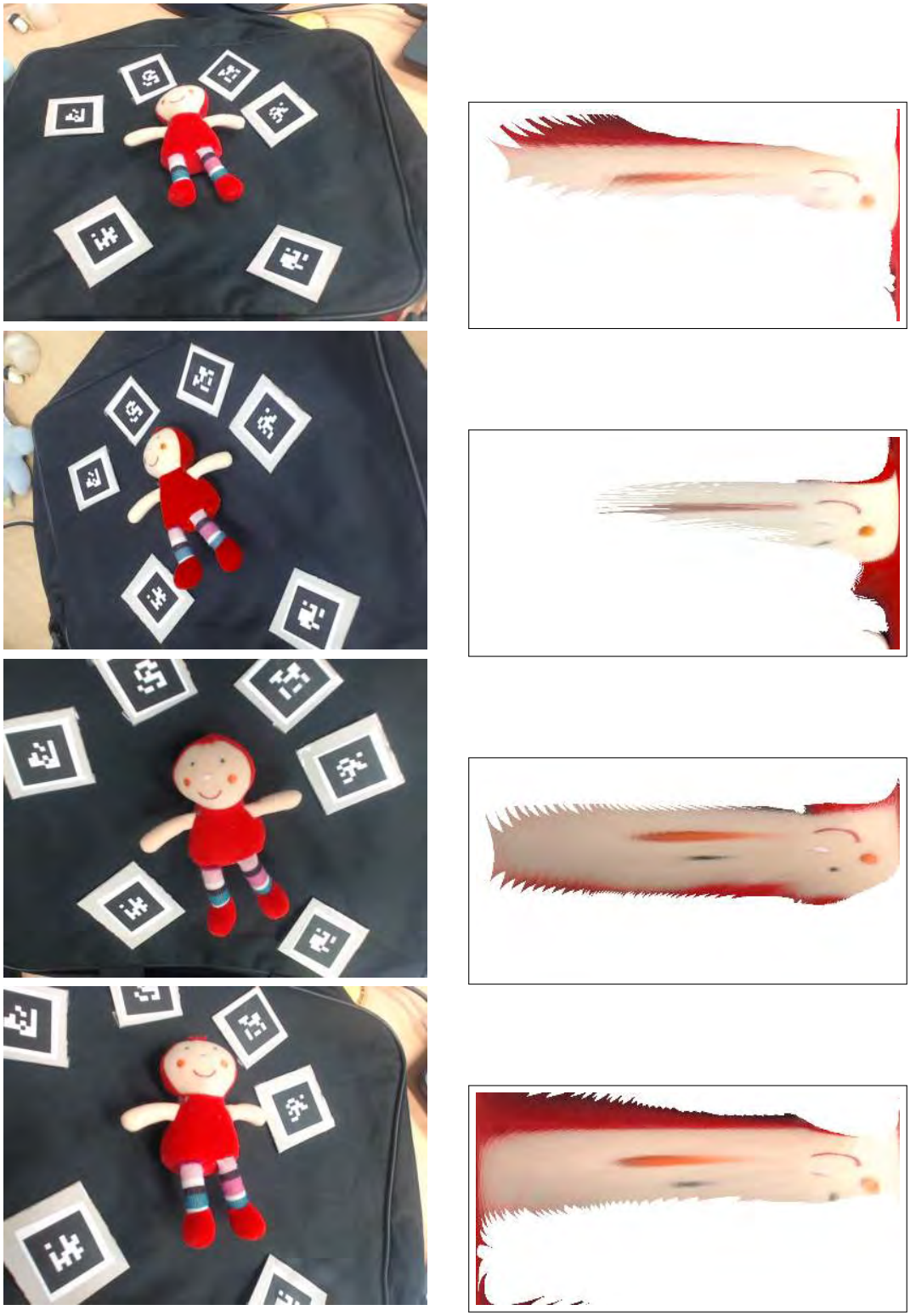


FIGURE 7.3 – Images de références crées pour une branche b (ici le visage de la marionnette) à partir d'un point de vue i. Chaque image I_i (première colonne) de la branche est projetée dans une image (partielle) I_i^b (deuxième colonne). L'image de texture finale I_b pour la branche peut être obtenue en fusionnant ces images comme décrit dans section 7.3.4.

l'image de référence I_i, et pour chaque pixel de I_i, on retient dans le tampon que le point $S(t, \theta)$ le plus proche de la caméra. Ainsi, chaque point de $S(t, \theta)$ visible sur I_i retient sa couleur dans l'image I_b. Ensuite, nous générons une carte de label, similaire à un deuxième tampon z, de sorte que chaque pixel de I_i reçoive le label de la surface canal la plus proche du point de vue, de la même façon que lorsque nous créons la carte de confiance au chapitre 4. Cependant, comme chaque objet reconstruit est une union de plusieurs surfaces canal, certaines surfaces canal sont partiellement à l'intérieur d'autres. Ainsi, ces surfaces cachées n'obtiennent une couleur d'aucune image, ce qui crée des zones à compléter dans la texture. Néanmoins, compléter toutes les surfaces cachées, indépendamment de la profondeur, est en fait inutile puisque cette surface est en 3D à l'intérieur d'une autre, et reste donc non visible. Pour attribuer une couleur à chaque surface de façon cohérente, on complète donc la couleur des points visibles et leur voisins uniquement, c'est à dire, à chaque point d'une surface visible et, pour plus de robustesse, aux points d'une surface cachée donc la profondeur est située à une distance inférieure à un seuil choisi ϵ de la surface occultante.

7.3.4 Texturation d'une branche à partir de plusieurs images

Comme discuté dans section 7.2, pour chaque point de la surface, il peut y avoir plusieurs images à partir desquelles la texture peut être sélectionnée. Dans notre approche, étant donné l'ensemble V des images I_i dans lesquelles un point de la surface est visible, nous appliquons la texture de la meilleure image de l'ensemble qui convient à la surface S. Pour cela, nous utilisons le critère de confiance défini dans le chapitre 4 et utilisé à l'origine pour l'*inpainting*.

Pour rappel, une carte de confiance $\mathcal{C}$ est construite à partir de ce critère pour évaluer la qualité de l'information projetée sur I_i. Elle donne, pour chaque pixel de l'image, un score de confiance basé sur deux paramètres : la distance entre le point de vue et le plan tangent de la surface où le pixel est projeté, et l'angle d'inclinaison entre la normale à la surface et l'axe de la caméra.

La valeur de la confiance diminue à mesure que la distance augmente et que l'angle d'inclinaison augmente. Des valeurs plus élevées de trust correspondent à des pixels de bonne qualité, alors que des valeurs plus faibles indiquent des points susceptibles de représenter des régions du plan éloignées de la caméra et/ou vues sous un angle très oblique ce qui arrive plus fréquemment que dans le cas planaire où l'angle reste constant sur un plan.

Sur la base de ce critère, pour chaque point de vue i, nous créons une carte de confiance $\mathcal{C}_i$ pour chaque pixel du domaine de l'image de référence $A \times [0, 2\pi]$ discrétisée.

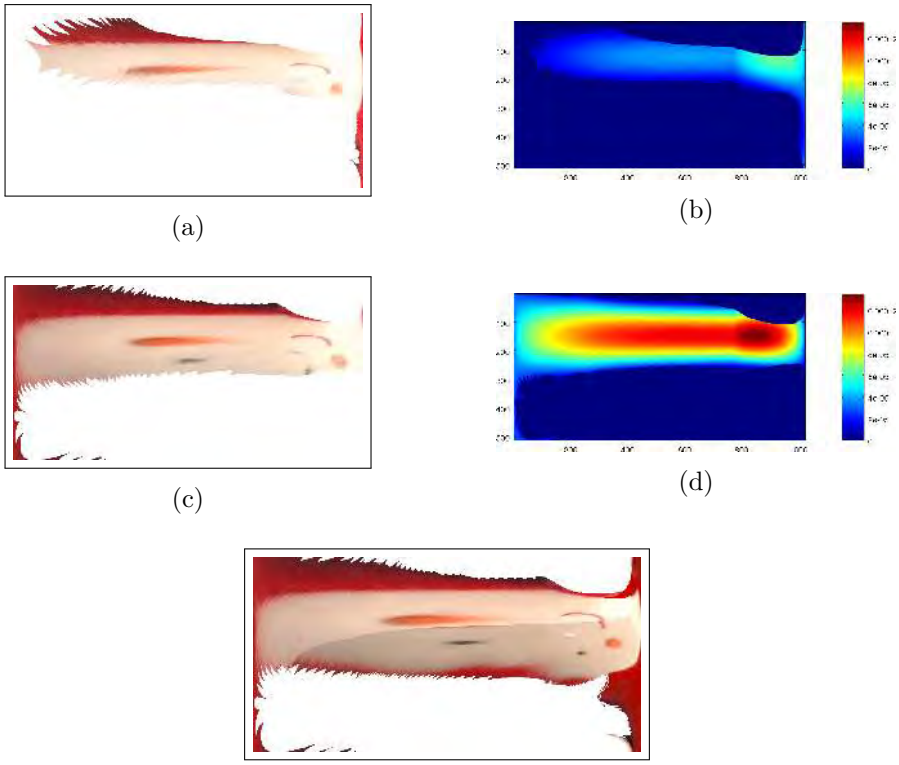
$$\mathcal{C}_i : \begin{cases} A \times [0, 2\pi] & \longrightarrow \mathbb{R} \\ p & \longmapsto \mathcal{C}_i(p) = \text{trust}(P_i) \end{cases} \tag{7.1}$$

avec $\mathcal{C}_i(p) = 0$ si p n'est pas une projection d'un point du plan Π. La figure 7.4 montre un exemple de deux cartes de confiance correspondant à deux points de vue distincts, la seconde image (c) ayant de meilleures valeurs de confiance que la première image (a).

Ensuite, nous construisons l'image de référence I_b de la branche b en sélectionnant les pixels ayant la plus grande valeur de confiance :

$$I_b(p) = I_i(p), \quad i = \text{argmax}_i \, \mathcal{C}_i(p). \tag{7.2}$$

L'image de sortie fusionne toutes les textures des différents points de vue. Cependant, comme le montre la figure 7.4, il peut y avoir des zones de l'image pour lesquelles aucune



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

FIGURE 7.4 – Carte de confiance de la branche b de la tête selon deux points de vue différents (sur quatre, les deux autres points de vue ne sont pas affichés). (a,c) : les images de référence I_i^b générées, chacune, à partir d'une image I_i. (b,d) : La carte de confiance correspondant où les valeurs les plus élevées (respectivement les plus faibles) sont peintes en rouge (respectivement en bleu). (e) l'image de référence finale I_b où nous avons fusionné les quatre points de vue.

texture ne peut être récupérée car elles ne sont visibles sur aucune des images d'entrée. De plus, des artefacts peuvent exister en raison des différentes expositions des images et de problèmes d'alignement. Nous abordons ces problèmes dans la section suivante.

7.4 Améliorations de la texture

Dans la section précédente, nous avons montré comment retrouver et appliquer les textures existantes dans le modèle géométrique, comment déterminer la branche correcte, et comment choisir la meilleure image en fonction de la géométrie. Comme indiqué dans la section 7.3, certains problèmes affectent encore la qualité des textures reconstruites. Tout d'abord, on peut rencontrer des problèmes d'alignements durant l'application des images dans l'image de référence en raison de l'approximation géométrique entre les points de vue. D'une part, notre modèle 3D ne fait qu'approximer implicitement la géométrie de l'objet réel par un ensemble de surfaces canal non dégénérées. D'autre part, une branche 3D est calculée par un processus de triangulation aux moindres carrés [Dur+16]. Pour ces deux raisons, la géométrie reconstruite est approximative et donc la texture n'est pas toujours exactement re-projetée dans l'espace des paramètres.

La reconstruction classique utilise une triangulation aux moindres carrés pour reconstruire un point 3D à partir de sa projection dans plusieurs images [HZ04]. Ici, le

même processus de triangulation est appliqué pour reconstruire une sphère, c'est-à-dire, un point 3D et un rayon. Les textures de différentes images peuvent avoir des discontinuités d'éclairage, en raison d'une exposition différente ou de conditions d'éclairage différentes. Enfin, certaines parties du modèle peuvent ne pas être visibles sur les images et la texture correspondante doit être générée. Les sections suivantes introduisent des améliorations sur les textures que nous avons mises en place pour surmonter ces problèmes.

7.4.1 Correction de l'exposition

Nous prenons en compte les différentes expositions des images en appliquant séquentiellement un ajustement local et un ajustement global. L'ajustement local est appliqué séparément pour chaque branche du modèle. Ensuite, une optimisation globale de l'exposition est calculée pour toutes les images.

Correction locale

Lors de la construction de l'image de référence d'une branche, différentes images peuvent avoir des expositions différentes, ce qui entraîne des discontinuités de couleur et de luminosité dans l'image finale (voir figure 7.6). Pour faire face à la différence d'exposition, nous appliquons un étalonnage radiométrique de l'exposition dérivé de Zhang *et al.* [ZCC16] comme expliqué dans le chapitre 5. Étant donné une branche b et son ensemble V d'images I_i dans laquelle b est visible, on génère un ensemble d'images I_i^b, paramétrées dans le même domaine $A \times [0, 2\pi]$ comme I_b. Notons qu'en général (sans considérer les erreurs d'alignement), chaque pixel $I_i^b(p)$ est le même pixel pour chaque i, c'est-à-dire qu'il représente le même point de la surface, éventuellement avec une couleur différente. Si, pour certaines images de V, la valeur de $I_i^b(p)$ n'est pas définie parce que le point n'est pas visible ou occulté, on fixe sa valeur à une valeur arbitraire de 0. Il faut maintenant fusionner ces images pour obtenir I_b tout en ajustant et corrigeant l'exposition. Nous formulons ce problème comme le problème d'optimisation non linéaire suivant :

$$\min_{\alpha_i, r(p)} \sum_{i,p} \Big(I_i^b(p) - \alpha_i \, L(p) \Big)^2 , \tag{7.3}$$

où α_i est le coefficient d'exposition pour I_i et $L(p)$ est la radiance des pixels. La figure 7.6 montre une comparaison visuelle entre l'exposition originale et l'exposition normalisée.

Correction globale

Une fois que les expositions dans les images $(I_i^b) \in V$ ont été corrigées pour chaque branche b indépendamment, nous ajustons l'exposition de l'objet 3D entier. Les pixels d'une même image I_i peuvent être projetés sur différentes branches, c'est-à-dire différents domaines $I_i^{b_1}$ et $I_i^{b_2}$ héritent de différents paramètres de correction d'exposition. Ainsi, l'exposition peut varier sur les parties adjacentes du modèle 3D. Pour éviter cette situation, nous ajustons les différentes corrections d'exposition de sorte que la variance soit minimisée sur une seule image I_i. Nous définissons pour les n images et les p branches la matrice $\mathtt{A}_{n \times p} = (\alpha_{i,j})$, où $\alpha_{i,j}$ est la correction d'exposition de la i-ième image dans la j-ième branche. Ensuite, pour chaque image I_i nous estimons un coefficient β_i qui minimise la variance du $\alpha_{i,j}$ sur toutes les branches j. Plus formellement, nous considérons une matrice diagonale $\mathbf{X} = \mathrm{diag}\,(\beta_1, \beta_2, \ldots, \beta_p)$ qui, multipliée par A, minimise la

(a)

(b)

(c)

FIGURE 7.5 – La correction de l'exposition et l'enregistrement sur une image de référence globale. (a) : l'image de référence d'une branche sans correction d'exposition montrant une discontinuité nette entre la texture de deux images. (b) : l'image de référence avec correction d'exposition menant à une transition plus douce entre les images (c) : l'image d'une branche avec correction d'enregistrement : la frontière entre sourire et la joue est maintenant continue.

variance ; nous voulons alors trouver l'ensemble des valeurs $\hat{\mathbf{X}}$ telles que :

$$\hat{\mathbf{X}} = \text{argmin}_{\mathbf{X}} \left(\sum_i \text{Var}((\mathtt{A}\,\mathbf{X})_i) \right), \tag{7.4}$$

où $(\mathbf{A}\,\mathbf{X})_i$ est la i-ième ligne de $\mathbf{A}\,\mathbf{X}$. Nous mettons ensuite à jour la texture de chaque *mesh* j avec le nouveau coefficient d'exposition $\hat{\alpha}_j = \hat{\beta}_j\,\alpha_j$ La figure 7.7 montre la différence entre un rendu sans la correction globale (gauche) et avec la correction globale (droite).

(a) (b)

FIGURE 7.6 – L'artefact causé par la différence d'exposition des images. (a) : la partie inférieure du visage de ce personnage est plus claire que la partie supérieure (b) : après correction de l'exposition, la correction locale conduit à une couleur cohérente entre les deux parties du visage, la correction globale fixe la différence de couleur entre les deux bras.

7.4.2 Correction de l'alignement

En raison de différences dans le modèle ou des erreurs d'étalonnage, un décalage dans l'alignement des paramétrisations des branches peut se produire et causer des discontinuités dans l'image de texture finale I_b, dans les régions où la texture provient de vues différentes avec des valeurs de confiance proches (comme illustré sur la figure 7.5b). Pour cela, nous considérons l'application de texture comme une minimisation de l'énergie par coupure de graphe [Ari+14]. Pour un pixel p de la texture de l'image I_b, nous attribuons un label l correspondant à la vue dont p a été pris (comme expliqué dans la section 7.3.3). L'énergie suivante est alors minimisée :

$$E = \sum_{p \in I_b} E_d(p, l) + \lambda \sum_{(p_1, p_2) \in \mathcal{N}} E_r(p_1, p_2, l_1, l_2), \tag{7.5}$$

où E_d est le terme d'attache aux données, E_r est le terme de régularisation et λ est un paramètre qui règle l'importance des deux fonctions d'énergie. Dans le terme de données, nous pénalisons les labels avec une mauvaise confiance pour le pixel p. Ainsi, nous définissons E_d comme :

$$E_d(p, l) = f(\mathcal{C}_l(p)), \tag{7.6}$$

(a) (b)

FIGURE 7.7 – Texture générée sans (a) et avec (b) correction globale de l'exposition. Les artefacts causés par les différences d'exposition sur chaque image sont traités par la correction globale.

où f est une fonction décroissante pour $\mathcal{C}_l$. Le terme de régularisation pénalise le choix des pixels voisins appartenant à des vues différentes, afin d'assurer la cohérence des pixels d'une même région. Nous avons considéré une fonction similaire à celle définie dans HE et SUN [HS12b] :

$$E_r(p_1, p_2, l_1, l_2) = \|I_{l_1}(p_1) - I_{l_2}(p_1)\| + \|I_{l_1}(p_2) - I_{l_2}(p_2)\| . \tag{7.7}$$

Nous utilisons l'image calculée comme expliquée dans la section 7.3.4 comme initialisation de l'étiquetage de chaque pixel p de I_b. La figure 7.5c montre la texture de la branche après la minimisation de l'énergie, les discontinuités le long de la bouche ont été fixées.

7.4.3 Complétion des texture par l'*inpainting*

Dans nos paramètres, la reconstruction ne nécessite qu'un nombre limité d'images, au contraire des pipelines MVS classiques. Puisque l'objet est modélisé par un ensemble de surfaces canal, deux à cinq images suffisent généralement pour reconstruire complètement le modèle. De plus, les images n'ont pas besoin de couvrir la totalité de l'objet, ce qui permet de reconstruire des parties de l'objet qui ne sont pas visibles. Ceci est particulièrement utile lorsque l'on capture, par exemple, des objets allongés sur un support. Cependant, ceci conduit à une limitation évidente dans la texturation : comme certaines parties de l'objet peuvent être cachées, quelque soit le point de vue, aucune information ne peut être récupérée pour la texture. Ainsi, certains pixels de l'image de référence I_b peuvent ne pas être colorés. Pour texturer l'objet entièrement, nous remplissons les régions manquantes de I_b par deux méthodes de remplissage selon la nature de la texture. La classification du chapitre 3 ne permet pas de distinguer les texture circulaires des autres textures ; l'utilisateur choisit, par branche, le type de texture présente.

Nous considérons deux types de textures : **les textures circulaires**, où la couleur d'un pixel $p(t, \theta)$ est indépendante de θ, et **les textures régulières**. Par exemple, sur la figure 7.6, les jambes de la poupée ont, dans l'objet original, un motif symétrique en rotation, donc de type circulaire. A l'opposé, l'arrière de la tête ou le visage ont une texture de type régulier. Dans les deux cas, les zones manquantes sont celles du dos, qui n'a pas été capturé par les images car l'objet était couché sur un plan. Dans les deux cas, nous appliquons l'algorithme d'*inpainting* basé sur *PatchMatch* [Bar+09], mais avec une étape d'initialisation différente de l'algorithme.

Dans le cas des pieds, et plus généralement pour les textures à symétrie de révolution, nous nous appuyons sur la symétrie de la texture pour l'*inpainting*. L'avantage de la symétrie est qu'elle résout les problèmes de décalage dans le sens du squelette. On peut forcer l'initialisation de sorte qu'elle suive la direction de l'axe de révolution. La figure 7.8 montre les différentes étapes de notre procédure pour une des jambes. A partir de l'image de référence originale I_b (figure 7.8a), nous créons une nouvelle image I_s, appelée l'image de *structure* (figure 7.8b) : pour chaque colonne de I_b, nous choisissons la valeur de pixel avec la confiance maximale sur la colonne, et nous attribuons cette valeur à la colonne entière. Notez que la structure de l'image peut encore avoir des régions non remplies si une colonne entière de I_b n'a aucune donnée. Nous utilisons ensuite l'algorithme de correspondance *PatchMatch* afin de créer une correspondance entre l'image de référence originale I_b et I_s : pour chaque *patch* de I_s nous trouvons le meilleur *patch* correspondant dans l'image originale. Nous appliquons la carte de correspondance calculée à I_s pour générer une nouvelle image qui a une texture plus similaire à celle d'origine, mais avec les mêmes régions non remplies. Nous appliquons ensuite l'algorithme classique de *PatchMatch* pour compléter les régions non remplies (figure 7.8c). La figure 7.9 montre la branche 3D avec et sans la finition de la texture.

Pour les textures régulières, comme par exemple l'arrière de la tête, nous utilisons la méthode de classification du chapitre 3 pour choisir l'approche d'*inpainting* appropriée.

7.5 Résultats expérimentaux

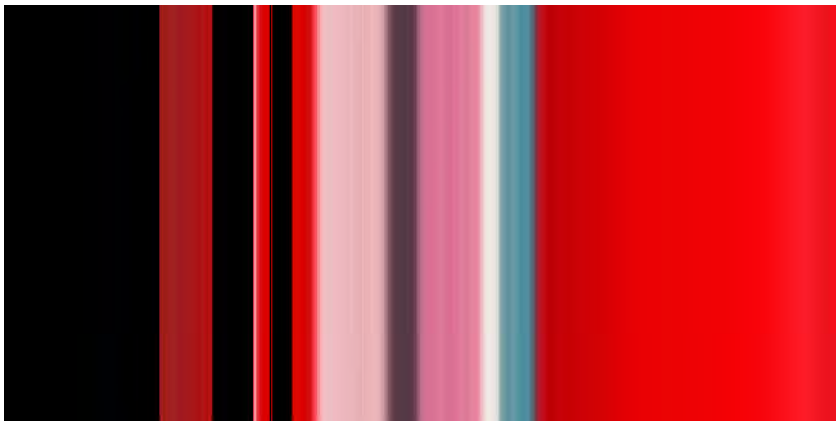
Dans cette section, nous présentons des résultats de notre approche. Nous énonçons d'abord les détails de notre implémentation. Nous affichons ensuite quelques résultats de rendu que nous commentons par rapport à aux méthodes existantes.

7.5.1 Détails d'implémentation

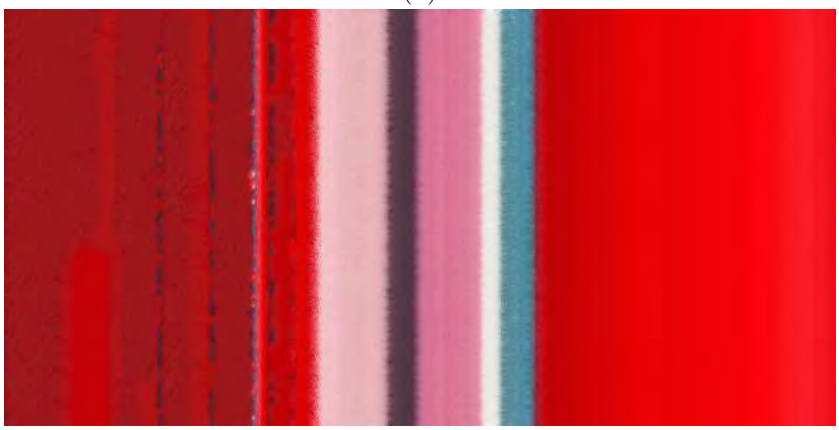
Une fois l'objet 3D reconstruit, nous utilisons la bibliothèque OpenCV pour créer les images de texture à partir des images acquises. Le problème d'optimisation de la correction d'exposition est résolu avec la bibliothèque Ceres Solver [A+16]. La finition de la texture est implémentée avec le *framework* de traitement d'image G'MIC [Tsc16]. L'utilisateur sélectionne pour chaque branche la méthode d'*inpainting* à utiliser. Pour chaque branche, la méthode d'*inpainting* est choisie automatiquement avec l'approche *a contrario* du chapitre 3.

La méthode de reconstruction et la méthode de texturation sont implémentées en C++. Le tableau 7.1 détaille le temps d'exécution pour des peluches, pour chaque étape de l'algorithme. La reconstruction basée sur le squelette est séparée en trois étapes principales : d'abord, la projection perspective du squelette est calculée sur chaque image, puis l'utilisateur associe les différentes extrémités et enfin, la triangulation est effectuée pour chaque branche. Avant d'extraire la texture des différentes images, un *z-buffer* est utilisé pour caractériser la surface du canal visible sur chaque pixel. Le temps de calcul du *z-buffering* pourrait être optimisé en effectuant ces calculs en parallèle sur le GPU. En moyenne, il faut 73 s pour la correction d'exposition, 120 s pour la correction d'enregistrement et 77 s pour la complétion de la texture. En moyenne, la phase de texturation prend 4 min pour notre implémentation basée CPU sur une machine Linux Mint équipée d'un CPU Intel i7-6700HQ à 2.6 GHz avec 8 GB de RAM.

(a)



(b)



(c)

FIGURE 7.8 – Complétion du maillage selon la structure de texture. (a) : l'image de référence d'entrée I_b. (b) : l'image de structure générée I_s. (c) : l'image finale complète (*cf.* figure 7.9 du rendu final).

7.5.2 Résultats

Les peluches sont de bons candidats car elles peuvent facilement être modélisées par un ensemble de surfaces canaux. Ils sont principalement remplies de mousse, qui est un matériel isotrope : cela crée des formes qui se dilatent de façon isotrope autour de l'axe, ce qui conduit à des surfaces circulaires autour de la direction de l'axe principal. Aucune

FIGURE 7.9 – texturation d'une des jambes : (à gauche) la texture originale sans *inpainting* et (à droite) le résultat du processus d'*inpainting* appliquée à la texture.

Peluche	Bleue	Souris	Rouge	Ours	Lapin
Estimation de la projection du squelette	1.7 s	1.4 s	1.7 s	1.8 s	1.8 s
Triangulation	1.3 s	2.4 s	1.3 s	1.4 s	1.6 s
Z-buffering (CPU)	29 s	39 s	27 s	26 s	28 s
Extraction de la texture des images	9.9 s	15.1 s	9.6 s	8.4 s	9.8 s
Correction de l'exposition	76 s	120 s	71 s	78 s	20 s
Correction de l'alignement	88 s	137 s	145 s	106 s	153 s
Complétion de la texture	58 s	128 s	89 s	51 s	57 s

TABLE 7.1 – Temps de calcul pour un ensemble de peluches.

hypothèse sur le nombre de pattes ou de queues n'est nécessaire.

La figure 7.10 montre des résultats sur le processus de complétion de la texture pour plusieurs branches. La première colonne montre pour une seule branche, l'image de référence lors de la projection de toutes les images sur cette dernière I_b sans aucune correction. Nous observons des discontinuités de luminosité dans la texture en raison de l'exposition différente des images. La deuxième colonne affiche la même image de référence après la correction de l'exposition et après le processus de complétion de la texture décrit dans section 7.4.3. Selon la branche et le type de texture, nous propageons la texture autour de l'axe de symétrie (deuxième et quatrième ligne) ou nous utilisons l'*inpainting* basé sur *PatchMatch*. La troisième colonne montre la branche 3D avec la texturation finale.

La figure 7.11 affiche, pour chaque ligne, le résultat final sur trois peluches différentes. Les deux premières colonnes montrent le modèle rendu sous deux points de vue différents avec la texture d'origine, sans aucun ajustement d'alignement et d'exposition. Dans la deuxième colonne, notons que le dos des peluches n'est pas texturé car elles reposaient sur un support lors de la capture, aucune image n'est disponible pour ces régions. De plus, les différences d'exposition et les désalignements de texture sont clairement visibles dans tous les modèles. Les deux dernières colonnes montrent le modèle texturé reconstruit : la plupart des discontinuités d'exposition ont été correctement traitées par notre algorithme d'ajustement et les textures sont plus lisses. Globalement, chaque modèle a une meilleure exposition globale et la structure des textures régulières est respectée (jambes, oreilles

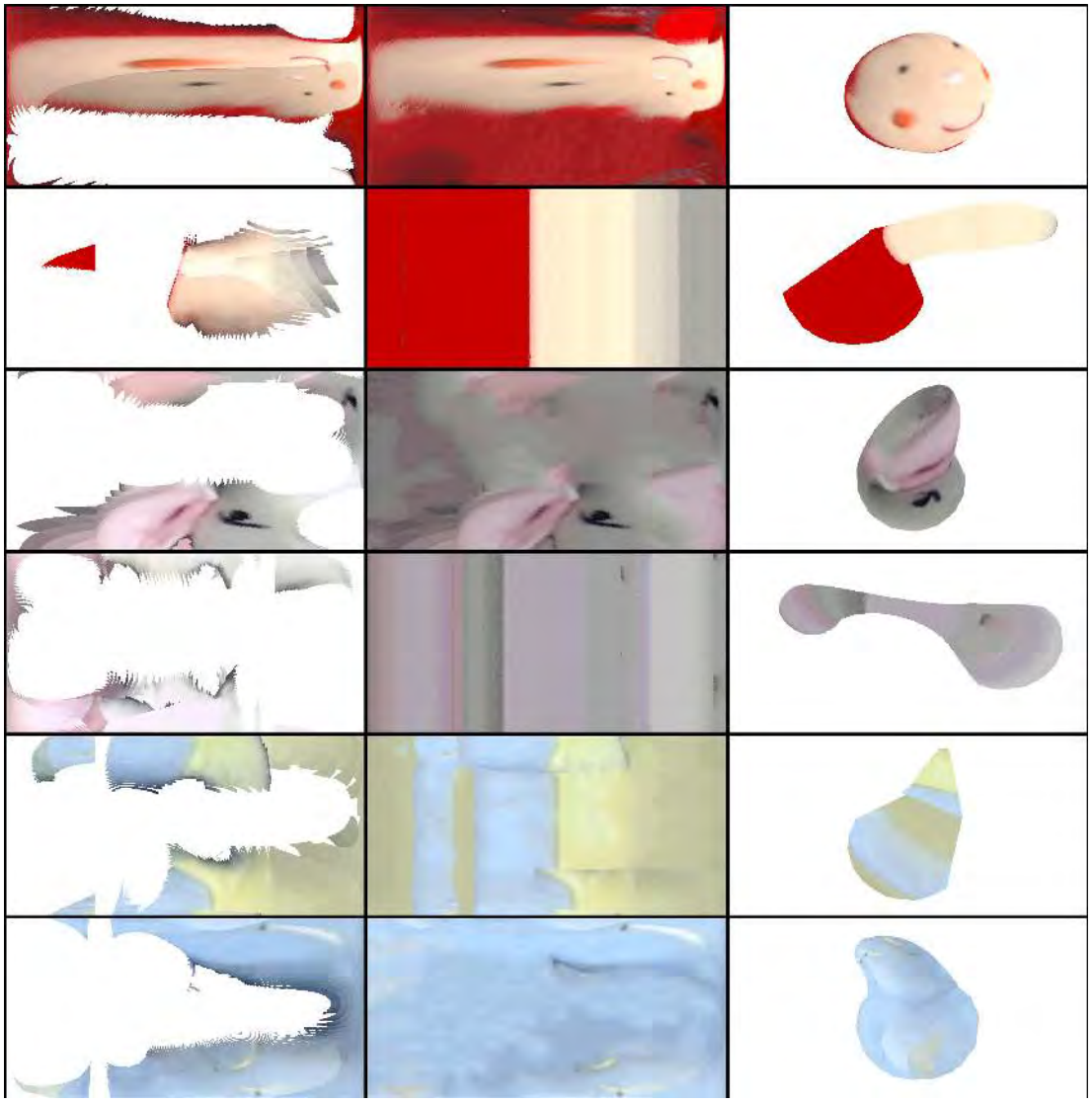


FIGURE 7.10 – Amélioration de la texture de plusieurs branches : la première colonne montre la texture originale telle que projetée sur l'image de référence I_b tandis que la deuxième colonne montre la texture complétée. La troisième colonne montre la texture complétée appliquée à la branche 3D correspondante.

des deuxième et troisième modèles). Des résultats supplémentaires avec les modèles 3D associés sont disponible en ligne sur la plateforme Sketchfab : le lecteur peut trouver les modèles 3D avec la texture originale [Fay+18b] et après avoir appliqué notre pipeline pour améliorer les textures [Fay+18a]. Nous fournissons des résultats supplémentaires : des modèles avec la texture originale [Fay+18b] et après avoir appliqué notre pipeline pour améliorer les textures [Fay+18a].

Il reste cependant des limitations ; celles-ci sont abordées dans la section suivante.

7.5.3 Limitations

La méthode de complétion est choisie manuellement par l'utilisateur en fonction du type de texture (régulière, symétrique en rotation, stochastique *etc.*). Afin d'automatiser la tâche, il serait intéressant d'analyser l'image de branche originale I_b afin de classer la

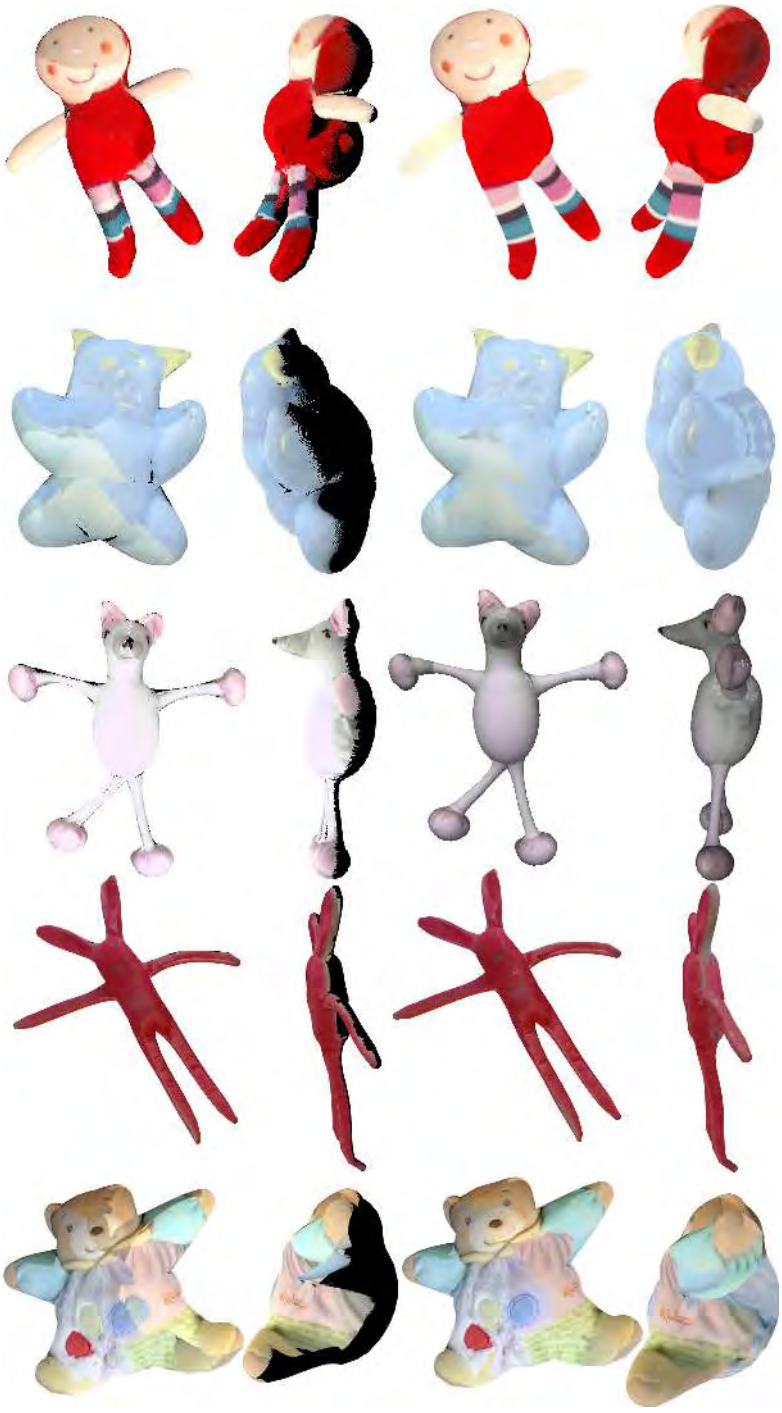


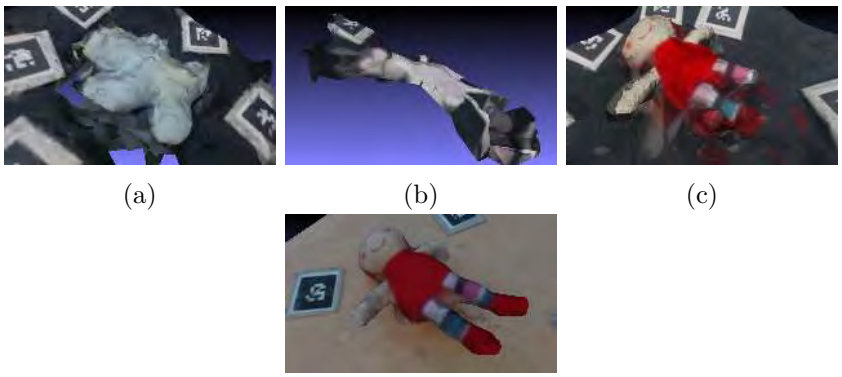
FIGURE 7.11 – Complétion de la texture sur cinq peluches. Les deux premières colonnes montrent le modèle brut avec une application directe de la texture. Les deux dernières colonnes montrent le modèle avec la texture améliorée par rapport à l'exposition et à l'alignement et complétée via l'*inpainting*. Les reconstructions des deux premières lignes ont utilisé 4 images en entrée, et celle de la troisième ligne (la souris) a utilisé trois images en entrée.

nature de la texture en circulaire ou régulière.

Notre méthode dépend de l'estimation précise des positions de la caméra pour avoir

(a) (b) (c)

FIGURE 7.12 – Un exemple d'objet (a) qui ne satisfait pas nos hypothèses : la tête est une surface qui ne peut pas être modélisée correctement comme surface canal, affectant ainsi le mappage de la texture ; la méthode décrite dans section 7.12b aide à obtenir une meilleure texture de l'objet (c).



(a) (b) (c)

(d)

FIGURE 7.13 – Quelques résultats obtenus avec le pipeline MVS open-source de la littérature, AliceVision [Ali17], utilisant nos images en entrée. Les objets mal texturés (a) ou les objets avec des parties minces (b) sont difficiles à reconstruire depuis quelques images avec des pipelines MVS classiques. L'augmentation du nombre d'images et une meilleure couverture de l'espace autour de l'objet améliorent la qualité de la reconstruction finale : (c) et (d) ont été reconstruites à partir de 4 et de 30 images respectivement. D'autres exemples 3D de reconstruction MVS pour notre jeu de données d'objets sont disponibles ici [Fay+18c].

une reconstruction fiable. Un calibrage incorrect ou médiocre peut affecter les résultats, en particulier lors de la projection de la texture de la branche sur l'image de référence I_b. De plus, la méthode de reconstruction suppose que l'objet est composé d'un ensemble de surfaces canal et qu'il est basé sur une triangulation aux moindres carrés, qui est sensible au bruit. La reconstruction d'objets, qui ne satisfont pas cette contrainte, peut être inexacte. La figure 7.12 montre un exemple de peluche qui ne satisfait pas la géométrie tubulaire, surtout au niveau de la tête qui n'est pas une surface canal typique (et qui aurait un axe médian en surface). Même si la reconstruction donne un modèle complet et satisfaisant, le décalage par rapport à la forme réelle affecte l'application de la texturation (figure 7.12b). Néanmoins, grâce à la correction du problème d'alignement décrite dans section 7.4.2 nous pouvons obtenir des meilleurs résultats pour la

texturation (figure 7.12c).

Concernant la correction d'exposition expliqué dans le chapitre 5, nous ne travaillons que sur l'éclairage : la peluche bleue sur figure 7.11 (deuxième ligne) a différentes variations de bleu selon l'image. Notre correction n'est pas capable d'unifier ou de lisser les différentes couleurs. Ces couleurs restent donc inchangées.

La figure 7.13 montre les résultats de reconstruction obtenus avec le pipeline MVS open-source AliceVision [Ali17] avec nos images en entrée. Une comparaison directe avec nos résultats n'est pas pertinente, car les hypothèses sont différentes : notre travail se limite aux objets tubulaires pour lesquels nous reconstruisons un modèle paramétrique texturé et complet qui se rapproche de la surface réelle, tandis que les approches MVS reconstruisent des objets plus généraux par triangulation, obtenant ainsi, en général, des modèles plus fins concernant les détails géométriques. Notre méthode a l'avantage de reconstruire, à partir de très peu d'images, un modèle texturé. Celui-ci est d'une qualité suffisante pour servir de base, par exemple, par un graphiste.

Comme le processus de reconstruction ne repose pas sur la recherche de correspondances d'images, notre méthode est capable de traiter des surfaces uniformes en couleurs ce que ne font pas les méthodes de reconstruction classique.

7.6 Conclusion

Dans ce chapitre, nous avons proposé une méthode de texturation et d'*inpainting* pour les modèles reconstruits comme un ensemble de surfaces canaux. La nature paramétrique du modèle conduit à de bons résultats concernant la texturation, et nous sommes en mesure de corriger l'exposition et le désalignement, et de reconstruire les informations de texture manquantes. À partir de quelques images d'une forme tubulaire 3D, un modèle 3D est reconstruit et peut être texturé par la méthode proposée.

Publication associée

Julien Fayer *et al.*. « Texturing and inpainting a complete tubular 3D object reconstructed from partial views ». In : *Computers & Graphics* 74 (2018), p. 126–136. ISSN : 0097-8493. DOI : https://doi.org/10.1016/j.cag.2018.05.012

Chapitre 8

SwapUp : Une application de Réalité Diminuée mobile temps réel pour l'édition d'environnement d'intérieur

Sommaire

Dans ce chapitre, nous allons aborder une deuxième application à finalité industrielle. Nous présentons *SwapUp* qui est une application développée par l'entreprise Innersense. *SwapUp* utilise les dernières technologies de Réalité Augmentée en matière de relocalisation. De plus, les travaux présentés dans les chapitres précédents sont intégrés pour effectuer un scénario de Réalité Diminuée d'une pièce d'intérieur.

Comme expliqué dans le chapitre 1, il n'existe pas d'applications grand public qui proposent de faire de l'aménagement temps-réel de pièces d'intérieur. Les applications sont actuellement limitées à la visualisation de meubles virtuels ajoutés grâce à la Réalité Augmentée. Elles ne sont pas capables d'effacer de manière interactive des objets réels et d'afficher en temps réel un rendu de la pièce sans les meubles. Actuellement au stade de démo, le but de *SwapUp* est de montrer qu'Innersense est capable dans un avenir proche, d'effacer des éléments réels de l'environnement et de fournir un rendu réaliste de l'environnement altéré. L'application a été développée, dans sa première version, entre le milieu de l'année 2018 et le début de l'année 2019. Actuellement, *SwapUp* construit un modèle géométrique de la pièce d'intérieur. L'utilisateur sélectionne, de manière interactive, les éléments à effacer. Une fois les objets sélectionnés, l'application calcule un

rendu photo-réaliste où les objets ont été effacés. L'utilisateur peut ensuite ajouter un meuble virtuel issu d'une base de données et l'afficher en temps réel dans la pièce, à la place d'un objet effacé.
L'objectif de ce chapitre est d'aborder l'architecture de l'application ainsi que les contributions scientifiques présentées dans les chapitres précédents et intégrées dans cette application. Nous commençons par présenter le processus global de *SwapUp*. Nous détaillons ensuite chacune des étapes qui composent le processus. Nous finissons enfin par une conclusion qui aborde les perspectives envisagées pour la suite du développement de *SwapUp*.

8.1 Présentation du processus global

Nous présentons rapidement les étapes majeures qui composent l'application *SwapUp*. *SwapUp* est divisé en plusieurs briques (voir figure 8.1) :

Étape d'acquisition. Cette première phase consiste à acquérir des informations de l'environnement (premier bloc bleu en partant du haut de la figure 8.1). Le *scan* et le suivi sont assurés par ARKit [App17]. Premièrement, l'utilisateur scanne la scène pour reconstruire un modèle géométrique de la pièce. Pour cela, il identifie dans *SwapUp* les plans qui forment la pièce. Ensuite, il définit les zones 3D à effacer dans l'environnement. Cela permet de créer les masques 3D des éléments qui seront effacés par la suite. Enfin, il effectue des captures de la pièce qui contiennent des informations nécessaires, notamment au niveau de la texture, pour la brique de diminution. Cette phase est optionnelle si nous avons déjà construit le modèle géométrique de notre environnement. Il suffit alors de scanner une zone de la pièce pour permettre à *SwapUp* de resituer le modèle 3D par rapport à l'environnement réel.

Étape de diminution. Cette brique se charge de récupérer les données créées durant la phase d'acquisition (images, plans, caméras). Le processus de diminution est ensuite effectué pour obtenir des textures complétées pour chaque plan majeur de la scène 3D (murs, sol). Le modèle 3D de la pièce est alors complété avec les textures appliquées sur chaque plan. Cette étape est également optionnelle si la pièce a déjà été acquise et complétée lors d'un enregistrement passé. Notons que cette étape ne se fait pas en temps réel car la complétion des zones masquées met un temps non négligeable à s'effectuer.

Étape de rendu. Comme l'étape précédent s'est effectuée hors-ligne, *SwapUp* commence par se relocaliser. La masque 2D est ensuite calculé à partir de la zone d'effacement 3D visible dans la *frame* courante. Ensuite, le modèle 3D complété précédemment est projeté dans le flux vidéo en remplaçant les pixels du masque de la zone d'effacement dans l'image. L'utilisateur peut ensuite ajouter des meubles virtuels dans le rendu de la scène d'intérieur.

8.2 Acquisition de la pièce et des objectifs utilisateur

Dans cette section, nous abordons comment les différentes informations nécessaires à la diminution et à l'application du rendu sont récupérées et traitées.

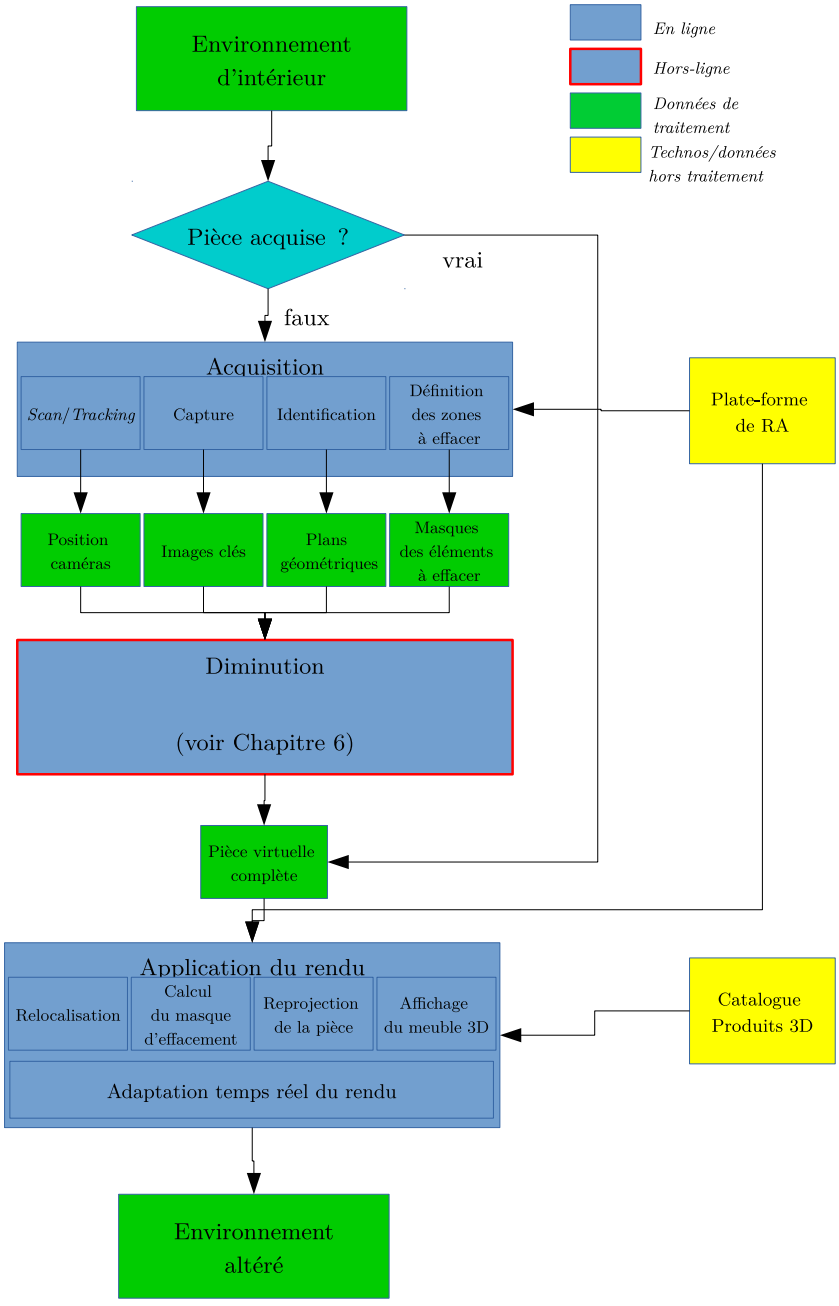


FIGURE 8.1 – Processus global de *SwapUp*. À partir d'un environnement d'intérieur, l'utilisateur construit, si c'est la première utilisation, un modèle virtuel de la pièce d'intérieur. Cette construction est divisée en deux étapes. Une première étape, appelée étape d'acquisition où l'application construit un modèle géométrique de la pièce à partir des informations 3D obtenues par ARKit [App17]. Ensuite, la deuxième étape, appelée étape de diminution, complète les plans du modèle avec la texture obtenue dans plusieurs images clés. Enfin, le modèle 3D texturé est affiché dans le flux de la caméra à la place des objets réels que l'on veut effacer. On a donc un environnement altéré où l'on peut ajouter des meubles virtuels à la place des meubles réels effacés.

FIGURE 8.2 – *scan* et sélection des plans. Les zones bleues correspondent aux zones où ARKit détecte suffisamment de points d'intérêt pour générer des plans locaux. L'utilisateur est invité à sélectionner une zone par plan, en commençant par le plan du sol. Une fois sélectionné, nous retournons le plan local de la zone qui remplit au mieux la contrainte de surface et la contrainte géométrique.

8.2.1 *Scan* et suivi de la pièce

Pour cette étape nous nous appuyons sur ARKit [App17], une nouvelle interface de programmation d'application (API) développée par Apple pour les dispositifs iOS. Cette interface est mise à disposition des développeurs pour faciliter la réalisation des applications de Réalité Augmentée. En particulier, ARKit utilise une technique appelée odométrie visuelle-inertielle [Leu+15] qui combine l'information provenant du capteur de mouvement inertiel du dispositif avec l'analyse de la scène visible par la caméra du dispositif. À partir du flux vidéo de la caméra de l'appareil, des points d'intérêt sont extraits des images. Ces points d'intérêt sont ensuite suivis sur plusieurs *frames*. Cela permet d'avoir des informations 3D comme la pose caméra courante et la position de ces points d'intérêt dans l'espace, et donc une reconstruction 3D éparse de la scène. Ces informations sont raffinées durant le *scan* de la pièce. Cette recherche perd cependant en efficacité si l'environnement est peu éclairé ou s'il y a peu de texture. Concernant la détection des plans, une fois un nombre minimal de points d'intérêt atteint, ces derniers sont regroupés pour former des plans locaux de sorte que l'association soit la meilleure en d'orientation et de position. Ensuite, les plans locaux qui ont les mêmes caractéristiques sont fusionnés. ARKit nous fournit donc l'ensemble des plans construits au sein d'une zone (coloriée en bleu dans la figure 8.2)
À partir de là, nous ne voulons garder que les plans représentant au mieux le sol et les murs. Pour le sol, nous récupérons le plan horizontal Δ_{sol} qui a la hauteur la plus basse dans le repère monde. Pour chaque mur, nous ne gardons que le plan qui satisfait au mieux deux contraintes : une contrainte de surface (il doit y avoir suffisamment de points appartenant à ce plan) et une contrainte géométrique (le plan doit être orthogonal au plan du sol). La figure 8.2 montre la récupération des plans du sol et d'un mur. Chaque plan sélectionné est représenté par un cercle appliqué à celui-ci.

8.2.2 Création des masques 3D

Une fois les plans récupérés, un masque 3D englobant l'élément à effacer est construit par l'utilisateur de manière interactive (voir figure 8.3).

FIGURE 8.3 – Création du masque 3D. L'application guide l'utilisateur durant la création du masque 3D sous la forme d'un cube. Premièrement, l'utilisateur sélectionne un coin du cube qui appartient au sol. Ensuite, l'utilisateur délimite le masque selon les trois axes de direction des arêtes du cube. D'abord la délimitation se fait selon les deux axes horizontaux (un axe puis un autre) puis se finit avec l'axe vertical.

8.2.3 Capture des image clés

Ensuite, l'utilisateur est invité à prendre des captures de la pièce (images clés) qui seront utilisés durant la prochaine phase, dite de diminution, pour compléter les parties cachées par les objets à enlever. Il est très important que ces images contiennent suffisamment d'information, notamment concernant les textures des plans pour pouvoir lancer le processus d'*inpainting*. Si, au moment de l'écriture de cette thèse, cette opération est complètement axée sur l'utilisateur ; l'une des perspectives est d'automatiser et de guider la capture de photos en associant un score à chaque *frame* qui représente la quantité et la qualité d'information contenue dans de l'image. Une piste pour construire ce score serait de reprendre le critère de confiance défini dans le chapitre 4 et de comparer et de sélectionner les meilleures images à partir de ce critère.

8.2.4 Détermination des délimitations des plans

À l'issue de l'étape de *scan* de la pièce, nous disposons d'un ensemble $\mathcal{P} = \{\Delta_i\}$ de plans géométriques Δ_i. Pour identifier les plans qui effectivement correspondent aux murs de la pièce, nous travaillons sous l'hypothèse que la scène vérifie le "monde de Manhattan" [CY99] : nous considérons que le plan sol Δ_{sol} et tous le autres plans qui soient perpendiculaires à Δ_{sol} et perpendiculaires ou parallèles entre eux. Nous représentons ces plan via un point du plan $\mathbf{P}_i = (x_i, y_i, z_i)^T$ et un vecteur normal $\overrightarrow{\mathbf{n}}_i = (a_i, b_i, c_i)^T$. Nous considérons également l'équation cartésienne d'un plan Δ_i pour tout point $\mathbf{M} = (x, y, z) \in \mathbb{R}^3$:

$$a_i x + b_i y + c_i z + d_i = 0 \tag{8.1}$$

avec $d_i = -\left(\overrightarrow{\mathbf{OP}}_i | \overrightarrow{\mathbf{n_i}}\right)$.

Premièrement, nous définissons les limites de chaque plan (ou coins) via les hypothèses fortes définies précédemment. Nous rajoutons l'hypothèse que les plans de la pièce forme un espace clos. En particulier, nous ne souhaitons garder que les points à l'intérieur de la pièce et supposons que la pièce est bordée de tout coté par un plan.

Les coins du plan du sol sont d'abord calculés. Pour cela, nous résolvons le système linéaire :

$$\mathtt{A}\,\mathbf{x} = \mathbf{b} \tag{8.2}$$

où

$$\mathtt{A}^{\,T} = \left(\vec{\mathbf{n}}_{\text{sol}}^{\,T}, \vec{\mathbf{n}}_{i}^{\,T}, \vec{\mathbf{n}}_{j}^{\,T}\right), \quad \mathbf{b} = (-d_{\text{sol}}, -d_i, -d_j)^T \tag{8.3}$$

pour les combinaisons $(\vec{\mathbf{n}}_{\text{sol}}, \vec{\mathbf{n}}_i, \vec{\mathbf{n}}_j)$ où $\vec{\mathbf{n}}_i$ et $\vec{\mathbf{n}}_j$ sont orthogonaux. Le système admettant une unique solution pour chaque combinaison, nous nous retrouvons avec N coins pour le plan du sol où N est le nombre de plans.

Nous notons $E_{\Delta_{\text{sol}}}$ l'ensemble des coins du sol trouvés. Nous devons maintenant calculer les coins des plans des murs que nous ne connaissons pas, c'est-à-dire ceux situés au plafond. Soit h la hauteur de la pièce, fournie actuellement par l'utilisateur. Pour chaque plan Δ_{mur} qui représente un mur, nous récupérons l'ensemble $E_{\Delta_{\text{mur}} \cap \Delta_{\text{sol}}}$ des coins du plan du sol qui intersectent le mur :

$$E_{\Delta_{\text{mur}} \cap \Delta_{\text{sol}}} = \{\mathbf{c} \in E_{\Delta_{\text{sol}}}, c \in \Delta_{\text{mur}}\} \tag{8.4}$$

Nous effectuons ensuite la translation $\mathbf{t} = h\vec{\mathbf{y}}$ à chaque coin de $E_{\Delta_{\text{mur}} \cap \Delta_{\text{sol}}}$ pour obtenir un nouveau coin, situé dans le plan du plafond de la pièce. Nous obtenons finalement, pour chaque plan Δ_{mur}, l'ensemble $E_{\Delta_{\text{mur}}}$ des coins qui le délimitent.

8.3 Diminution de la scène

Cette brique prend comme entrées les données suivantes : les images clés, les masques des objets à effacer dans chaque image clé, les plans géométriques, leurs coins ainsi que les poses caméra.

Premièrement, pour chaque plan, nous construisons son image rectifiée associée. Nous calculons pour cela, pour chaque image clé enregistrée précédemment, l'homographie qui associe le plan-image au plan géométrique. La zone appartenant au plan est ensuite transformée grâce à cette homographie.

Nous nous retrouvons donc avec une image rectifiée partielle, dans laquelle il manque probablement de l'information, soit à cause de l'obstruction d'un objet, soit à cause de l'absence d'acquisition d'information dans cette partie du plan. L'image rectifiée est ensuite divisée en deux images : la première est appelée carte de texture et la deuxième est appelée carte de luminosité. Chacune a sa zone masquée complétée par des procédés de complétion adaptés : une méthode basée motifs pour la carte de texture (*cf.* chapitre 3) et une méthode basée diffusion pour la carte de luminosité (*cf.* chapitre 5). Les deux cartes complétées sont enfin fusionnées pour aboutir à une image rectifiée complétée pour le plan considéré.

Une fois cette boucle effectuée pour chaque plan de la pièce, nous nous retrouvons avec un modèle 3D complet de la pièce qui peut ensuite être projeté dans l'environnement réel grâce à la brique suivante.

8.4 Projection du résultat sur la scène

Nous abordons dans cette section comment le modèle construit à la fin de la brique précédente est utilisé pour effectuer un rendu de la scène diminuée.

FIGURE 8.4 – De gauche à droite et de haut en bas : *frame* courante, masque 3D projeté dans la rame courante, le modèle 3D virtuel complété vu du même point de vue que la *frame* courante, la *frame* courante et en superposition, la projection du modèle 3D virtuel dans la zone du masque.

8.4.1 Relocalisation

La relocalisation est assuré par ARKit. Il suffit de scanner une partie la pièce pour resituer la caméra de l'appareil mobile dans l'environnement de la pièce.

8.4.2 Affichage de la pièce virtuelle

Comme on connaît la position de la caméra dans l'environnement, on peut appliquer la pièce virtuellement reconstruite dans le flux vidéo. Maintenant, nous devons récupérer la partie complétée par la brique de diminution dans le rendu purement virtuel et l'injecter dans le rendu de l'environnement réel.

L'image en bas à gauche de la figure 8.4 montre la reprojection de la pièce virtuelle vidée par rapport à la pose caméra de la rame courante. Chaque *frame* a deux zones : la zone connue $O \backslash \Omega$ et la zone masquée Ω, représentée par la projection 2D du masque 3D (voir l'image en haut, à droite de la figure 8.4). La zone masquée de la *frame* courante est remplacée par la projection du modèle 3D (voir l'image en bas à droite de la figure 8.4).

Une fois les pixels changés, il est nécessaire de corriger l'exposition des pixels changés par rapport à la rame courante. Si cela n'est pas fait, des incohérences au niveau de l'exposition de la caméra subsistent. Pour cela, nous appliquons l'algorithme de *Poisson blending* [PGB03] entre la *frame* courante et la zone de l'image modifiée par le masque. La figure 8.5 montre un exemple où l'exposition de la zone complétée a été corrigée. On constate un très bonne assimilation de la zone complétée dans la *frame* courante ce qui renforce le rendu photo-réaliste.

Concernant les temps de calcul de l'étape de diminution, nous avons intégré un para-

FIGURE 8.5 – Exemple de correction de l'exposition de la zone complétée par un processus de *Poisson blending*. Gauche : avant traitement. Droite : après traitement.

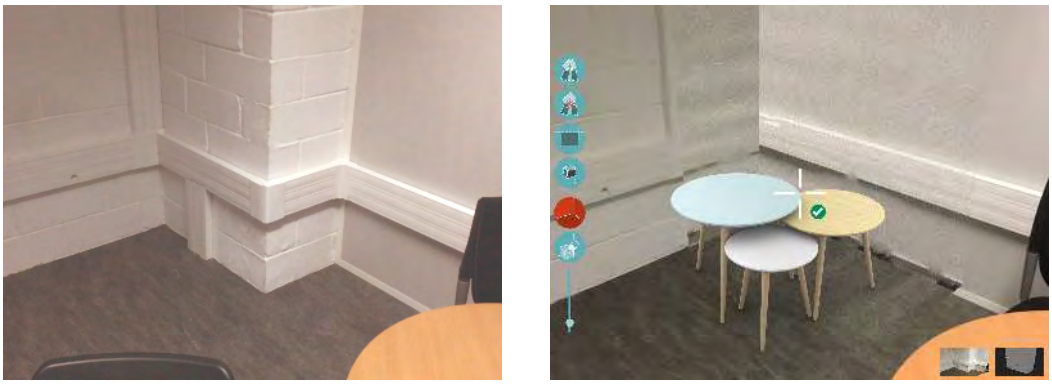


FIGURE 8.6 – Ajout d'un meuble virtuel dans la zone effacée. Gauche : avant l'effacement. Droite : après l'effacement. Dans cette situation, nous avons effectué une opération physiquement coûteuse. En effet, nous avons supprimé l'ex-croissance de la pièce située au centre de l'image. L'espace laissé vide est comblé par un modèle 3D de table.

mètre de qualité. Plus ce paramètre est faible, plus le temps de calcul est faible également. Généralement, avec un masque dont l'aire représente 30% de l'image, l'étape de diminution met 5 s a être exécutée pour une qualité minimale tandis qu'il faut 30 s pour un résultat en qualité pleine. Vu la durée d'exécution, la diminution pourrait être faite en temps réel mais le résultat ne serait pas de qualité, sauf si l'on veut remplacer un objet réel effacé par un objet virtuel. Un autre point qui n'est pas actuellement temps réel concerne le *Poisson blending*. Ce sont ces deux points de notre pipeline qui méritent une optimisation afin de coupler une utilisation temps réel et un rendu de qualité raisonnable.

8.4.3 Ajout de meubles virtuels

Une fois la *frame* courante corrigée avec le modèle virtuel, il est possible d'ajouter un meuble virtuel d'une banque de données dans la scène (*cf*. figure 8.6). Il pourra ainsi être affiché en 3D à la place d'un objet réel effacé.

8.5 Conclusion

Ce chapitre a présenté un exemple d'application de Réalité Diminuée qui intègre les travaux effectués durant cette thèse. L'objectif de cette application est d'effacer des meubles existants pour les remplacer des meubles virtuels venant d'une base de données. Cette application utilise les dernières technologies de Réalité Augmentée comme ARKit. Elle permet, à partir de ces technologies, de mettre en pratique les travaux scientifiques présentés durant les chapitres précédents, notamment en terme de complétion, de luminosité et de résolution de données.

Plusieurs perspectives sont possibles. La première est une intégration du processus de *Poisson blending* dans l'étape de rendu visuel pour gérer en temps réel la correction de l'exposition du modèle 3D projeté dans la *frame* courant. La deuxième perspective envisagée est de gérer des scènes d'intérieur plus complexes dont la structure de la pièce n'est pas un parallélépipède rectangle.

Enfin, il est prévu de tenir compte des capteurs 3D actuellement disponibles et qui peuvent être démocratisés au grand public d'ici une année. En effet, ARCore [Goo17] et ARKit [App17] sont des technologies qui se basent uniquement sur une caméra RGB. L'utilisation de capteurs 3D permettra d'avoir une géométrie beaucoup plus fournie. Nous pourrons aussi disposer d'un modèle 3D plus précis. Les briques d'acquisition et de projection seront modifiées pour tenir compte de cette ajout. On pourra avoir, pour chaque meuble, un maillage 3D au lieu d'un masque 3D. Il sera alors possible de déplacer le meuble dans la scène réelle. Pour cela, une idée est de partir de la preuve de concept évoquée durant la section 1.2.2 du chapitre 1. Leur utilisation dans une application grand public, par contre, est liée à leur démocratisation et implantation sur la prochaine génération de dispositifs mobiles, prévision toujours difficile à établir.

Conclusion

Nous résumons dans cette conclusion les contributions détaillées durant cette thèse ainsi que les perspectives possibles.

Contributions

Dans le chapitre 1, nous avons vu défini la Réalité Diminuée à travers le prisme de la vision par ordinateur. Nous avons listé les approches qui se rapprochent de cette définition. Pour réaliser une application de Réalité Diminuée dans un cadre d'environnement d'intérieur, nous avons conclu que :

— L'effacement d'un objet dans une scène implique la complétion de la texture des plans, occultée par cet objet d'où l'utilisation de la technique de l'*inpainting*.
— Un modèle géométrique 3D de la pièce est nécessaire pour mieux guider le processus d'*inpainting*.

Nous avons mis en évidence les limites de l'existant, à savoir le mauvais respect de la structure de la texture, la mauvaise propagation de la luminosité dans les zones occultées ainsi que le problème de résolution de l'*inpainting* sur les images rectifiées.

Pour cela, dans le chapitre 3, nous avons proposé une méthode de classification basée sur une approche *a contrario* qui classe les textures selon la structure et la distribution du motif afin de choisir la méthode d'*inpainting* qui complète au mieux cette texture. Pour les textures ambiguës, nous avons proposé une approche qui combine deux méthodes d'*inpainting* de l'état de l'art : l'approche basée *PatchMatch* [Bar+09] et l'approche par *offsets* [HS12b] en contrôlant l'initialisation en basse résolution.

Ensuite, dans le chapitre 4, nous avons proposé un critère de confiance radiométrique pour contrôler la propagation des données de basse qualité dans le masque de l'image rectifiée ; ce qui conduit à des zones floues dans le rendu final. Nous avons d'ailleurs proposé trois applications de ce critère : deux pour améliorer les deux méthodes d'*inpainting* utilisées ci-dessus et une dernière pour contrôler les discontinuités de résolution après complétion.

Dans le chapitre 5, nous avons adapté les méthodes de correction d'exposition au cas de l'*inpainting*. Nous avons ensuite proposé une méthode pour mieux guider l'*inpainting* dans les cartes de luminosité en complétant les courbes isophotes.

Grâce à ces contributions, nous avons pu développer des applications à portée industrielle. Nous avons vu dans le chapitre 6 deux applications de Réalité Diminuée qui vident une pièce à partir de quelques prises de vue et d'un modèle géométrique reconstruit.

Le chapitre 7 propose une extension de l'*inpainting* à la géométrie non-planaire en complétant la texture d'objets reconstruits par des surfaces canal.

Enfin, dans le chapitre 8, nous avons présenté l'application *SwapUp* qui met en pratique un scénario de Réalité Diminuée temps-réel. Celle-ci sert de base pour le développement de futurs produits chez Innersense visant à améliorer l'expérience d'aménagement d'intérieur en proposant l'effacement d'objets réels en plus de l'ajout d'objets virtuels déjà existant.

Perspectives scientifiques

Nous avons évoqués au fur et à mesure de cette thèse plusieurs perspectives. Premièrement, nous avons parlé d'une meilleure analyse des *offsets* dans les cas d'échec évoqués dans le chapitre 3 en analysant l'image sur plusieurs bandes de fréquence, ou encore en adaptant les critères de sélection des *offsets*.
Dans les applications évoquées dans le chapitre 6, les perspectives sont d'utiliser l'inférence statistique pour déterminer de façon plus rigoureuse si la quantité d'information est suffisante pour enclencher le processus d'*inpainting* ou s'il est nécessaire d'utiliser sur une synthèse de texture.
Le renforcement de la gestion de la luminosité est une perspective prévue également. Elle est actuellement considérée comme un problème de traitement d'image dans le chapitre 5. Créer un vrai modèle de luminosité en 3D renforcerait de manière significative le photo-réalisme.

Perspectives industrielles

Dans le chapitre 8, nous prévoyons de rendre l'application plus robuste. Pour cela, nous devrons gérer des environnement plus complexes (qui n'ont pas juste une forme parallélépipédique). Nous devrons également optimiser le code qui exécute l'étape de diminution ainsi que l'étape de *Poisson blending*.

Comme énoncé dans le chapitre 1, il y reste des problématiques non abordées durant la thèse pour développer une application permettant au grand public d'effectuer, de manière confortable, d'aménager son espace d'intérieur Une première problématique concerne l'interaction utilisateur. Celui-ci doit sélectionner manuellement les éléments à effacer en traçant les contours du masque d'effacement. Cela peut gêner l'expérience utilisateur. En effet, l'utilisateur doit, par exemple dans *SwapUp*, tourner autour de l'objet pour construire un masque 3D. Il faut donc avoir de l'espace pour permettre à l'utilisateur de construire le masque. Ensuite, il n'existe pas de modélisation sémantique dans le modèle 3D. Une deuxième perspective prévue est de pouvoir enrichir la scène avec une représentation sémantique. Cela facilitera le déplacement des objets réels. et permettrait d'envisager des interactions entre objets de la scène, potentiellement jusqu'aux interactions entre objets réels et/ou virtuels. Ces deux perspectives associées augmenteraient considérablement l'expérience utilisateur : ce dernier n'aurait qu'à pointer un objet dans le rendu de l'application, et le déplacer comme un objet virtuel.
L'arrivée des capteurs 3D "grand-publics", comme celui annoncé par Apple [App19], peut aider à réaliser ces perspectives. Les capteurs 3D vont permettre de construire un maillage dense de la pièce. Nous pourrons alors développer des méthodes, en s'inspirant notamment des approches de Silberman *et al.* [Sil+12] et de Wong *et al.* [WCM15],

pour générer le modèle sémantique de la pièce. Enfin, le développement des approches de *Deep-Learning* ont eu un impact dans la communauté scientifique. Il sera intéressant d'analyser ces méthodes, notamment pour guider la segmentation et créer la représentation sémantique.
Nous aurons finalement une application de Réalité Diminuée capable d'effectuer le fameux "Couper/Coller" d'objets dans un espace d'intérieur ; et permettre à Innersense de fournir une expérience d'aménagement virtuelle d'intérieur inédite.

Liste des publications

[Fay+18e] Julien FAYER *et al.*. « Radiometric confidence criterion for patch-based inpainting ». In : *Proceedings of the 24th International Conference on Pattern Recognition (ICPR 2018)*. Août 2018, p. 2723–2728. DOI : 10.1109/ICPR.2018.8545350

[Fay+18d] Julien FAYER *et al.*. « Texturing and inpainting a complete tubular 3D object reconstructed from partial views ». In : *Computers & Graphics* 74 (2018), p. 126–136. ISSN : 0097-8493. DOI : https://doi.org/10.1016/j.cag.2018.05.012

[Fay+17] Julien FAYER *et al.*. « Critère de confiance géométrique pour l'Inpainting basée patch (short paper) ». francais. In : *Journées francophones des jeunes chercheurs en vision par ordinateur (ORASIS), Colleville-sur-Mer, France, 12/06/2017-16/06/2017*. https ://hal.archives-ouvertes.fr : HAL, 2017, (en ligne). URL : http://oatao.univ-toulouse.fr/19266/

[Fay+16] Julien FAYER *et al.*. « Réalité diminuée : "couper/coller" interactif pour l'aménagement d'intérieur ». In : *Journées Françaises d'Informatique Graphique (JFIG 2016), Grenoble*. Déc. 2016. URL : https://jfayer.eu/site/pdf/jfig2016.pdf

Table des figures

Bibliographie

[A+16] Sameer Agarwal, Keir Mierle *et al.*. *Ceres Solver*. http://ceres-solver.org. 2016.

[Aga+04] Aseem Agarwala *et al.*. « Interactive Digital Photomontage ». In : *ACM Trans. Graph.* 23.3 (août 2004), p. 294–302. ISSN : 0730-0301. DOI : 10.1145/1015706.1015718. URL : http://doi.acm.org/10.1145/1015706.1015718.

[Ali17] AliceVision. *Photogrammetric Computer Vision Framework*. 2017. URL : https://alicevision.github.io/.

[Ami] Amikasa. *Amikasa App*. URL : http://amikasa.com/.

[App19] Apple. *Annonce du capteur 3D dans les Iphones en 2020*. 2019. URL : https://www.lesnumeriques.com/telephone-portable/apple-capteur-3d-inedit-pour-iphone-2020-n83453.html.

[App17] Apple Inc. *ARKit*. https://developer.apple.com/arkit/. 2017.

[AS09] Paul A. Ardis et Amit Singhal. « Visual salience metrics for image inpainting ». In : *Proceedings SPIE 7257, Visual Communications and Image Processing 2009*. T. 7257. 2009, p. 1–9. DOI : 10.1117/12.808942. URL : https://doi.org/10.1117/12.808942.

[Ari+14] M. Arikan, R. Preiner, C. Scheiblauer, S. Jeschke et M. Wimmer. « Large-Scale Point-Cloud Visualization through Localized Textured Surface Reconstruction ». In : *IEEE Transactions on Visualization and Computer Graphics* 20.9 (sept. 2014), p. 1280–1292. ISSN : 1077-2626. DOI : 10.1109/TVCG.2014.2312011.

[Art] Artefacto. *Offres d'Artefacto*. URL : https://www.artefacto-ar.com/offre/building/.

[Azu97] Ronald T. Azuma. « A Survey of Augmented Reality ». In : *Presence : Teleoperators and Virtual Environments* 6.4 (1997), p. 355–385. DOI : 10.1162/pres.1997.6.4.355. eprint : https://doi.org/10.1162/pres.1997.6.4.355. URL : https://doi.org/10.1162/pres.1997.6.4.355.

[Bar+09] Connelly Barnes, Eli Shechtman, Adam Finkelstein et Dan B Goldman. « PatchMatch : A Randomized Correspondence Algorithm for Structural Image Editing ». In : *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28.3 (août 2009). DOI : 10.1145/1531326.1531330. URL : http://doi.acm.org/10.1145/1531326.1531330.

[BMR01] F. Bernardini, I.M. Martin et H. Rushmeier. « High-quality texture reconstruction from multiple scans ». In : *IEEE Transactions on Visualization and Computer Graphics* 7.4 (2001), p. 318–332. ISSN : 10772626. DOI : 10.1109/2945.965346.

[Ber+00] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles et Coloma Ballester. « Image Inpainting ». In : *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. New York, NY, USA : ACM Press/Addison-Wesley Publ. Co., 2000, p. 417–424. ISBN : 1-58113-208-5. DOI : 10.1145/344779.344972. URL : http://dx.doi.org/10.1145/344779.344972.

[BKR17] Sai Bi, Nima Khademi Kalantari et Ravi Ramamoorthi. « Patch-based optimization for image-based texture mapping ». In : *ACM Transactions on Graphics* 36.4 (juil. 2017), p. 1–11. ISSN : 07300301. DOI : 10.1145/3072959.3073610.

[BRR11] Michael Bleyer, Christoph Rhemann et Carsten Rother. « PatchMatch Stereo-Stereo Matching with Slanted Support Windows. » In : *Proceedings of the British Machine Vision Conference (BMVC)*. T. 11. 2011, p. 1–11.

[BM07] Folkmar Bornemann et Tom März. « Fast Image Inpainting Based on Coherence Transport ». In : *Journal of Mathematical Imaging and Vision* 28.3 (juil. 2007), p. 259–278. ISSN : 1573-7683. DOI : 10.1007/s10851-007-0017-6. URL : https://doi.org/10.1007/s10851-007-0017-6.

[BK04] Y. Boykov et V. Kolmogorov. « An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9 (sept. 2004), p. 1124–1137. ISSN : 0162-8828. DOI : 10.1109/TPAMI.2004.60. URL : http://doi.org/10.1109/TPAMI.2004.60.

[BVZ01] Y. Boykov, O. Veksler et R. Zabih. « Fast approximate energy minimization via graph cuts ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.11 (2001), p. 1222–1239. ISSN : 01628828. DOI : 10.1109/34.969114. URL : http://doi.org/10.1109/34.969114.

[Buy+15] P. Buyssens, M. Daisy, D. Tschumperle et O. Lezoray. « Exemplar-Based Inpainting : Technical Review and New Heuristics for Better Geometric Reconstructions ». In : *IEEE Transactions on Image Processing* 24.6 (juin 2015), p. 1809–1824. ISSN : 1057-7149. DOI : 10.1109/TIP.2015.2411437. URL : http://doi.org/10.1109/TIP.2015.2411437.

[CG06] Vincenzo Caglioti et Alessandro Giusti. « Reconstruction of Canal Surfaces from Single Images Under Exact Perspective ». In : *Proceedings of the 2006 European Conference on Computer Vision (ECCV 2006)*. 2006, p. 289–300. DOI : 10.1007/11744023_23.

[CCS08] Jian-Feng Cai, Raymond H. Chan et Zuowei Shen. « A framelet-based image inpainting algorithm ». In : *Applied and Computational Harmonic Analysis* 24.2 (2008). Special Issue on Mathematical Imaging – Part II, p. 131–149. ISSN : 1063-5203. DOI : https://doi.org/10.1016/j.acha.2007.10.002. URL : http://www.sciencedirect.com/science/article/pii/S106352030700108X.

[Cal+08] M. Callieri, P. Cignoni, M. Corsini et R. Scopigno. « Masked photo blending : Mapping dense photographic data set on high-resolution sampled 3D models ». In : *Computers & Graphics* 32.4 (août 2008), p. 464–473. ISSN : 00978493. DOI : 10.1016/j.cag.2008.05.004.

[CS01a] Tony F. Chan et Jianhong Shen. « Nontexture Inpainting by Curvature-Driven Diffusions ». In : *Journal of Visual Communication and Image Representation* 12.4 (2001), p. 436–449. ISSN : 1047-3203. DOI : 10.1006/jvci.2001.0487. URL : http://dx.doi.org/10.1006/jvci.2001.0487.

[CS01b] Tony F. Chan et Jianhong Shen. « Nontexture Inpainting by Curvature-Driven Diffusions ». In : *Journal of Visual Communication and Image Representation* 12.4 (2001), p. 436–449. ISSN : 1047-3203. DOI : 10.1006/jvci.2001.0487.

[CSZ06] Tony F. Chan, Jianhong Shen et Hao-Min Zhou. « Total Variation Wavelet Inpainting ». In : *Journal of Mathematical Imaging and Vision* 25.1 (juil. 2006), p. 107–125. ISSN : 1573-7683. DOI : 10.1007/s10851-006-5257-3. URL : https://doi.org/10.1007/s10851-006-5257-3.

[Che+13] Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu et Daniel Cohen-Or. « 3Sweep : Extracting Editable Objects from a Single Photo ». In : *ACM Trans. Graph.* 32.6 (nov. 2013), 195 :1–195 :10. ISSN : 0730-0301. DOI : 10.1145/2508363.2508378.

[CT14] Guillaume Chican et Mohamed Tamaazousti. « Constrained PatchMatch for Image Completion ». In : *Proceedings of the 10th International Symposium on Advances in Visual Computing ISVC*. Las Vegas, NV, USA : Springer International Publishing, déc. 2014. DOI : 10.1007/978-3-319-14249-4_53.

[Cho17] F. Chollet. *Deep Learning with Python*. Manning Publications, 2017. ISBN : 9781617294433. URL : https://blog.keras.io/the-limitations-of-deep-learning.html.

[Cor+13] M. Corsini, M. Dellepiane, F. Ganovelli, R. Gherardi, A. Fusiello et R. Scopigno. « Fully Automatic Registration of Image Sets on Approximate Geometry ». In : *International Journal of Computer Vision* 102.1-3 (mar. 2013), p. 91–111. ISSN : 0920-5691. DOI : 10.1007/s11263-012-0552-5.

[CY99] James M. Coughlan et A. L. Yuille. « Manhattan World : Compass Direction from a Single Image by Bayesian Inference ». In : *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*. ICCV '99. Washington, DC, USA : IEEE Computer Society, 1999, p. 941–. ISBN : 0-7695-0164-8. URL : http://dl.acm.org/citation.cfm?id=850924.851554.

[CPT03] A. Criminisi, P. Perez et K. Toyama. « Object removal by exemplar-based inpainting ». In : *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* T. 2. Juin 2003, p. 721–728. DOI : 10.1109/CVPR.2003.1211538. URL : https://doi.org/10.1109/CVPR.2003.1211538.

[DTL13] Maxime Daisy, David Tschumperlé et Olivier Lézoray. « A Fast Spatial Patch Blending Algorithm for Artefact Reduction in Pattern-based Image Inpainting ». In : *SIGGRAPH Asia 2013 Technical Briefs*. SA '13. Hong Kong, Hong Kong : ACM, 2013, 8 :1–8 :4. ISBN : 978-1-4503-2629-2. DOI : 10.1145/2542355.2542365. URL : http://doi.acm.org/10.1145/2542355.2542365.

[DMM00] Agnès Desolneux, Lionel Moisan et Jean-Michel Morel. « Meaningful Alignments ». In : *International Journal of Computer Vision* 40.1 (oct. 2000), p. 7–23. issn : 1573-1405. doi : 10.1023/A:1026593302236. url : https://doi.org/10.1023/A:1026593302236.

[Dur+15] Bastien Durix, Geraldine Morin, Sylvie Chambon, Celine Roudet et Lionel Garnier. « Towards Skeleton Based Reconstruction : From Projective Skeletonization to Canal Surface Estimation ». In : *Proceedings of the IEEE International Conference on 3D Vision*. 2015, p. 545–553. doi : 10.1109/3DV.2015.67.

[Dur+16] Bastien Durix, Geraldine Morin, Sylvie Chambon, Celine Roudet et Lionel Garnier. « Skeleton-based Multiview Reconstruction ». In : *Proceedings of the IEEE International Conference on Image Processing*. 2016, p. 4047–4051. doi : {10.1109/ICIP.2016.7533120}.

[EL99] Alexei A Efros et Thomas K Leung. « Texture synthesis by non-parametric sampling ». In : *iccv*. IEEE. 1999, p. 1033.

[Ela+05] M. Elad, J.-L. Starck, P. Querre et D.L. Donoho. « Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA) ». In : *Applied and Computational Harmonic Analysis* 19.3 (2005). Computational Harmonic Analysis - Part 1, p. 340–358. issn : 1063-5203. doi : https://doi.org/10.1016/j.acha.2005.03.005. url : http://www.sciencedirect.com/science/article/pii/S1063520305000655.

[Fay+18a] Julien Fayer, Bastien Durix, Simone Gasparini et Géraldine Morin. *3D textured models with inpainting*. https://skfb.ly/6yCCE. 2018.

[Fay+18b] Julien Fayer, Bastien Durix, Simone Gasparini et Géraldine Morin. *3D textured models without inpainting*. https://skfb.ly/6yCCG. 2018.

[Fay+18c] Julien Fayer, Bastien Durix, Simone Gasparini et Géraldine Morin. *MVS reconstruction with AliceVision*. https://skfb.ly/6yxQO. 2018.

[Fay+18d] Julien Fayer, Bastien Durix, Simone Gasparini et Géraldine Morin. « Texturing and inpainting a complete tubular 3D object reconstructed from partial views ». In : *Computers & Graphics* 74 (2018), p. 126–136. issn : 0097-8493. doi : https://doi.org/10.1016/j.cag.2018.05.012.

[Fay+16] Julien Fayer, Simone Gasparini, Géraldine Morin et Benjamin Coudrin. « Réalité diminuée : "couper/coller" interactif pour l'aménagement d'intérieur ». In : *Journées Françaises d'Informatique Graphique (JFIG 2016), Grenoble*. Déc. 2016. url : https://jfayer.eu/site/pdf/jfig2016.pdf.

[Fay+17] Julien Fayer, Simone Gasparini, Géraldine Morin et Maxime Daisy. « Critère de confiance géométrique pour l'Inpainting basée patch (short paper) ». francais. In : *Journées francophones des jeunes chercheurs en vision par ordinateur (ORASIS), Colleville-sur-Mer, France, 12/06/2017-16/06/2017*. https ://hal.archives-ouvertes.fr : HAL, 2017, (en ligne). url : http://oatao.univ-toulouse.fr/19266/.

[Fay+18e] Julien Fayer, Géraldine Morin, Simone Gasparini, Maxime Daisy et Benjamin Coudrin. « Radiometric confidence criterion for patch-based inpainting ». In : *Proceedings of the 24th International Conference on Pattern Recognition (ICPR 2018)*. Août 2018, p. 2723–2728. doi : 10.1109/ICPR.2018.8545350.

[FU99] Manfred Feil et Andreas Uhl. « Real-time image analysis using wavelets : the “a trous” algorithm on MIMD architectures ». In : t. 3645. SPIE, 1999, doi : 10.1117/12.343799. url : https://doi.org/10.1117/12.343799.

[FH15] Yasutaka Furukawa et Carlos Hernández. « Multi-View Stereo : A Tutorial ». In : *Foundations and Trends® in Computer Graphics and Vision* 9.1-2 (2015), p. 1–148. issn : 1572-2740. doi : 10.1561/0600000052.

[GDA13] Ignacio Garcia-Dorado, Ilke Demir et Daniel G Aliaga. « Automatic urban modeling using volumetric reconstruction with surface graph cuts ». In : *Computers & Graphics* 37.7 (nov. 2013), p. 896–910. issn : 00978493. doi : 10.1016/j.cag.2013.07.003.

[Goh18] Philippe-Antoine Gohard. « De la réalité augmentée sans marqueurs pour l'aménagement d'intérieur à la réalité diminuée sur plateforme mobile ». Thèse de doct. Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), 2018.

[Gol10] D. B. Goldman. « Vignette and Exposure Calibration and Compensation ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.12 (déc. 2010), p. 2276–2288. issn : 0162-8828. doi : 10.1109/TPAMI.2010.55.

[Goo17] Google LLC. *ARCore*. https://developers.google.com/ar/. 2017.

[Gro14] Rafael Grompone von Gioi. *A Contrario Line Segment Detection*. Sous la dir. de Computer Science. Springer, New York, NY, 2014. doi : https://doi.org/10.1007/978-1-4939-0575-1.

[Gui+13] Christine Guillemot, Mehmet Turkan, Olivier Le Meur et Mounira Ebdelli. « Object removal and loss concealment using neighbor embedding methods ». In : *Signal Processing : Image Communication* 28.10 (2013), p. 1405–1419. issn : 0923-5965. doi : https://doi.org/10.1016/j.image.2013.08.020. url : http://www.sciencedirect.com/science/article/pii/S0923596513001379.

[HZ04] R. I. Hartley et A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN : 0521540518, 2004.

[HS12a] K. He et J. Sun. « Computing nearest-neighbor fields via Propagation-Assisted KD-Trees ». In : *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Juin 2012, p. 111–118. doi : 10.1109/CVPR.2012.6247665.

[HS12b] Kaiming He et Jian Sun. « Statistics of Patch Offsets for Image Completion ». In : *Proceedings of the 12th European Conference on Computer Vision - Volume Part II*. ECCV'12. Florence, Italy : Springer-Verlag, 2012, p. 16–29. isbn : 978-3-642-33708-6. doi : 10.1007/978-3-642-33709-3_2.

[HH05] Y. Hel-Or et H. Hel-Or. « Real-time pattern matching using projection kernels ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.9 (sept. 2005), p. 1430–1445. issn : 0162-8828. doi : 10.1109/TPAMI.2005.184.

[HB14] J. Herling et W. Broll. « High-Quality Real-Time Video Inpaintingwith PixMix ». In : *Visualization and Computer Graphics, IEEE Transactions on* 20.6 (juin 2014), p. 866–879. issn : 1077-2626. doi : 10.1109/TVCG.2014.2298016.

[HS04] Carlos Hernández Esteban et Francis Schmitt. « Silhouette and stereo fusion for 3D object modeling ». In : *Computer Vision and Image Understanding* 96.3 (déc. 2004), p. 367–392. ISSN : 10773142. DOI : 10.1016/j.cviu.2004.03.016.

[HC32] D Hilbert et S Cohn-Vossen. *Geometry and the Imagination.* New York : Chelsea : Chelsea Publishing Co., 1932. DOI : 10.1126/science.116.3023.643-a.

[HSL16] Yinlin Hu, Rui Song et Yunsong Li. « Efficient Coarse-To-Fine PatchMatch for Large Displacement Optical Flow ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Juin 2016.

[HH10] Hui-Yu Huang et Chun-Nan Hsiao. « A patch-based image inpainting based on structure consistence ». In : *Proceedings of the 2010 International Computer Symposium (ICS).* Déc. 2010, p. 165–170. DOI : 10.1109/COMPSYM.2010.5685527.

[Hua+13] J. Huang, J. Kopf, N. Ahuja et S. B. Kang. « Transformation guided image completion ». In : *Proceedings of the IEEE International Conference on Computational Photography (ICCP).* Avr. 2013, p. 1–9. DOI : 10.1109/ICCPhot.2013.6528313.

[Hua+14] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja et Johannes Kopf. « Image Completion Using Planar Structure Guidance ». In : *ACM Trans. Graph.* 33.4 (juil. 2014), 129 :1–129 :10. ISSN : 0730-0301. DOI : 10.1145/2601097.2601205. URL : http://doi.acm.org/10.1145/2601097.2601205.

[HG08] Q. Huynh-Thu et M. Ghanbari. « Scope of validity of PSNR in image/video quality assessment ». English. In : *Electronics Letters* 44 (13 juin 2008), 800–801(1). ISSN : 0013-5194. URL : https://digital-library.theiet.org/content/journals/10.1049/el_20080522.

[IKE] IKEA. *IKEA Place.* URL : https://www.ikea.com/ms/fr_FR/france/appli_IKEA/appli_ikea_place.html.

[Iso+18] Mariko Isogawa, Dan Mikami, Kosuke Takahashi, Daisuke Iwai, Kosuke Sato et Hideaki Kimata. « Which is the Better Inpainted Image ?Training Data Generation Without Any Manual Operations ». In : *International Journal of Computer Vision* (nov. 2018). ISSN : 1573-1405. DOI : 10.1007/s11263-018-1132-0. URL : https://doi.org/10.1007/s11263-018-1132-0.

[JHS10] Songkran Jarusirisawad, Takahide Hosokawa et Hideo Saito. « Diminished reality using plane-sweep algorithm with weakly-calibrated cameras ». In : *Progress in informatics* 7 (mar. 2010), p. 11–20. ISSN : 13498614. URL : http://ci.nii.ac.jp/naid/80021000360/.

[KSY15] N. Kawai, T. Sato et N. Yokoya. « Diminished Reality Based on Image Inpainting Considering Background Geometry ». In : *IEEE Transactions on Visualization and Computer Graphics* 22.99 (2015), p. 1–1. ISSN : 1077-2626. DOI : 10.1109/TVCG.2015.2462368. URL : http://dx.doi.org/10.1109/TVCG.2015.2462368.

[Kaw+13] Norihiko KAWAI, Masayoshi YAMASAKI, Tomokazu SATO et Naokazu YOKOYA. « [Paper] Diminished Reality for AR Marker Hiding Based on Image Inpainting with Reflection of Luminance Changes ». In : *ITE Transactions on Media Technology and Applications* 1.4 (2013), p. 343–353. DOI : 10.3169/mta.1.343.

[KY12] Norihiko KAWAI et Naoto YOKOYA. « Image inpainting considering symmetric patterns ». In : *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE. 2012, p. 2744–2747. URL : http://ieeexplore.ieee.org/document/6460733/.

[Kna+17] Arno KNAPITSCH, Jaesik PARK, Qian-Yi ZHOU et Vladlen KOLTUN. « Tanks and temples ». In : *ACM Transactions on Graphics* 36.4 (juil. 2017), p. 1–13. ISSN : 07300301. DOI : 10.1145/3072959.3073599.

[KZ04] V. KOLMOGOROV et R. ZABIH. « What energy functions can be minimized via graph cuts ? » In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2 (fév. 2004), p. 147–159. ISSN : 0162-8828. DOI : 10.1109/TPAMI.2004.1262177. URL : http://doi.org/10.1109/TPAMI.2004.1262177.

[Köp+15] M. KÖPPEL, M. Ben MAKHLOUF, K. MÜLLER et T. WIEGAND. « Fast image completion method using patch offset statistics ». In : *Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP)*. Sept. 2015, p. 1795–1799. DOI : 10.1109/ICIP.2015.7351110. URL : http://doi.org/10.1109/ICIP.2015.7351110.

[KAS10] O. KORKALO, M. AITTALA et S. SILTANEN. « Light-weight marker hiding for augmented reality ». In : *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*. Oct. 2010, p. 247–248. DOI : 10.1109/ISMAR.2010.5643590.

[Küm+11] R. KÜMMERLE, G. GRISETTI, H. STRASDAT, K. KONOLIGE et W. BURGARD. « G2o : A general framework for graph optimization ». In : *2011 IEEE International Conference on Robotics and Automation*. Mai 2011, p. 3607–3613. DOI : 10.1109/ICRA.2011.5979949.

[Kwa+03] Vivek KWATRA, Arno SCHÖDL, Irfan ESSA, Greg TURK et Aaron BOBICK. « Graphcut Textures : Image and Video Synthesis Using Graph Cuts ». In : *ACM Trans. Graph.* 22.3 (juil. 2003), p. 277–286. ISSN : 0730-0301. DOI : 10.1145/882262.882264. URL : http://doi.acm.org/10.1145/882262.882264.

[Lap18] P.A. LAPLANTE. *Encyclopedia of Image Processing*. CRC Press, 2018. ISBN : 9781351032735. URL : https://books.google.fr/books?id=jIqADwAAQBAJ.

[LGG11] O. LE MEUR, J. GAUTIER et C. GUILLEMOT. « Examplar-based inpainting based on local geometry ». In : *2011 18th IEEE International Conference on Image Processing*. Sept. 2011, p. 3401–3404. DOI : 10.1109/ICIP.2011.6116441.

[Leu+15] Stefan LEUTENEGGER, Simon LYNEN, Michael BOSSE, Roland SIEGWART et Paul FURGALE. « Keyframe-based visual–inertial odometry using nonlinear optimization ». In : *The International Journal of Robotics Research* 34.3 (2015), p. 314–334. DOI : 10.1177/0278364914554813. eprint : https://doi.org/10.1177/0278364914554813. URL : https://doi.org/10.1177/0278364914554813.

[Li+16] D. Li, T. Shao, H. Wu et K. Zhou. « Shape Completion from a Single RGBD Image ». In : *IEEE Transactions on Visualization and Computer Graphics* PP.99 (2016), p. 1–1. ISSN : 1077-2626. DOI : 10.1109/TVCG.2016.2553102.

[Li+13] Zhuwen Li, Yuxi Wang, Jiaming Guo, Loong-Fah Cheong et S.Z. Zhou. « Diminished reality using appearance and 3D geometry of internet photo collections ». In : *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. Oct. 2013, p. 11–19. DOI : 10.1109/ISMAR.2013.6671759.

[Lin+06] Wen-Chieh Lin, J. Hays, Chenyu Wu, Yanxi Liu et V. Kwatra. « Quantitative Evaluation of Near Regular Texture Synthesis Algorithms ». In : *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. T. 1. Juin 2006, p. 427–434. DOI : 10.1109/CVPR.2006.233.

[LRP17] Jose-Luis Lisani, Silvia Ramis et Francisco J. Perales. « A Contrario Detection of Faces : A Case Example ». In : *SIAM Journal on Imaging Sciences* 10.4 (jan. 2017), p. 2091–2118. DOI : 10.1137/17m1118774. URL : https://doi.org/10.1137/17m1118774.

[Liu+18] Jiaying Liu, Shuai Yang, Yuming Fang et Zongming Guo. « Structure-Guided Image Inpainting Using Homography Transformation ». In : *IEEE Transactions on Multimedia* 20.12 (2018).

[LL03] Yanxi Liu et Wen-Chieh Lin. « Deformable texture : the irregular-regular-irregular cycle ». In : (2003).

[LLH04] Yanxi Liu, Wen-Chieh Lin et James Hays. « Near-regular Texture Analysis and Manipulation ». In : *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. Los Angeles, California : ACM, 2004, p. 368–376. DOI : 10.1145/1186562.1015731. URL : http://doi.acm.org/10.1145/1186562.1015731.

[LC13] Yunqiang Liu et V. Caselles. « Exemplar-Based Image Inpainting Using Multiscale Graph Cuts ». In : *IEEE Transactions on Image Processing* 22.5 (mai 2013), p. 1699–1711. ISSN : 1057-7149. DOI : 10.1109/TIP.2012.2218828.

[Low+99] David G Lowe *et al.*. « Object recognition from local scale-invariant features. » In : *iccv*. T. 99. 2. 1999, p. 1150–1157.

[Lue+02] David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy et Amitabh Varshney. *Level of Detail for 3D Graphics*. New York, NY, USA : Elsevier Science Inc., 2002. ISBN : 1558608389.

[MES08] J. Mairal, M. Elad et G. Sapiro. « Sparse Representation for Color Image Restoration ». In : *IEEE Transactions on Image Processing* 17.1 (jan. 2008), p. 53–69. ISSN : 1057-7149. DOI : 10.1109/TIP.2007.911828.

[MV07] Ezio Malis et Manuel Vargas. *Deeper understanding of the homography decomposition for vision-based control*. Research Report RR-6303. INRIA, 2007, p. 90. URL : https://hal.inria.fr/inria-00174036.

[Man02] S Mann. « Mediated Reality with implementations for everyday life. Presence Connect ». In : *Presence : Teleoperators and Virtual Environments* (2002), p. 119–122.

[MRB12] R. Martínez-Noriega, A. Roumy et G. Blanchard. « Exemplar-based image inpainting : Fast priority and coherent nearest neighbor search ». In : *Proceedings of the 2012 IEEE International Workshop on Machine Learning for Signal Processing*. Sept. 2012, p. 1–6. DOI : 10.1109/MLSP.2012.6349810.

[M+98] Andrzej Materka, Michal Strzelecki *et al.*. « Texture analysis methods–a review ». In : *Technical university of lodz, institute of electronics, COST B11 report, Brussels* (1998), p. 9–11.

[MEG13] O. Le Meur, M. Ebdelli et C. Guillemot. « Hierarchical Super-Resolution-Based Inpainting ». In : *IEEE Transactions on Image Processing* 22.10 (oct. 2013), p. 3779–3790. ISSN : 1057-7149. DOI : 10.1109/TIP.2013.2261308.

[MK94] Paul Milgram et Fumio Kishino. « A Taxonomy of Mixed Reality Visual Displays ». In : *IEICE Transactions on Information Systems* E77-D.12 (déc. 1994). URL : http://vered.rose.utoronto.ca/people/paul%5C_dir/IEICE94/ieice.html.

[MIT02] MIT Media Lab. *VisionTexture (VisTex)*. 2002. URL : http://vismod.media.mit.edu/vismod/imagery/VisionTexture/.

[MT14] A. Morgand et M. Tamaazousti. « Generic and real-time detection of specular reflections in images ». In : *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP 2014)*. T. 1. Jan. 2014, p. 274–282. URL : https://ieeexplore.ieee.org/abstract/document/7294821.

[Mor+12] B. Morse, J. Howard, S. Cohen et B. Price. « PatchMatch-Based Content Completion of Stereo Image Pairs ». In : *Proceedings of the Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*. Oct. 2012, p. 555–562. DOI : 10.1109/3DIMPVT.2012.59.

[New+17] Alasdair Newson, Andrés Almansa, Yann Gousseau et Patrick Pérez. « Non-Local Patch-Based Image Inpainting ». In : *Image Processing On Line* 7 (2017), p. 373–385. DOI : 10.5201/ipol.2017.189.

[OH13] Takahiro Ogawa et Miki Haseyama. « Image inpainting based on sparse representations with a perceptual metric ». In : *EURASIP Journal on Advances in Signal Processing* 2013.1 (déc. 2013), p. 179. ISSN : 1687-6180. DOI : 10.1186/1687-6180-2013-179. URL : https://doi.org/10.1186/1687-6180-2013-179.

[Oja+02] T. Ojala, T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen et S. Huovinen. « Outex - new framework for empirical evaluation of texture analysis algorithms ». In : *Object recognition supported by user interaction for service robots*. T. 1. Août 2002, 701–706 vol.1. DOI : 10.1109/ICPR.2002.1044854.

[OT06] F. Ortiz et F. Torres. « Automatic detection and elimination of specular reflectance in color images by means of MS diagram and vector connected filters ». In : *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 36.5 (sept. 2006), p. 681–687. ISSN : 1094-6977. DOI : 10.1109/TSMCC.2005.855424.

[OL82] Mark H. Overmars et Jan van Leeuwen. « Dynamic multi-dimensional data structures based on quad- and k—d trees ». In : *Acta Informatica* 17.3 (août 1982), p. 267–285. ISSN : 1432-0525. DOI : 10.1007/BF00264354. URL : https://doi.org/10.1007/BF00264354.

[Pat13] Harshal S. Patil. « Study and Review of Various Image Texture Classification Methods ». In : 2013.

[PGG17] Viorica Patraucean, Pierre Gurdjos et Rafael Grompone von Gioi. « Joint A Contrario Ellipse and Line Detection ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (avr. 2017), p. 788–802. ISSN : 0162-8828. DOI : 10.1109/TPAMI.2016.2558150. URL : http://ieeexplore.ieee.org/document/7458912/.

[PGB03] Patrick Pérez, Michel Gangnet et Andrew Blake. « Poisson Image Editing ». In : *ACM Trans. Graph.* 22.3 (juil. 2003), p. 313–318. ISSN : 0730-0301. DOI : 10.1145/882262.882269. URL : http://doi.acm.org/10.1145/882262.882269.

[PP97] Martin Peternell et Helmut Potmann. « Computing Rational Parametrizations of Canal Surfaces ». In : *Journal of Symbolic Computation* 23.2-3 (fév. 1997), p. 255–266. DOI : 10.1006/jsco.1996.0087.

[Pho75] Bui Tuong Phong. « Illumination for Computer Generated Pictures ». In : *Commun. ACM* 18.6 (juin 1975), p. 311–317. ISSN : 0001-0782. DOI : 10.1145/360825.360839. URL : http://doi.acm.org/10.1145/360825.360839.

[QFR18] Glen Queguiner, Matthieu Fradet et Mohammad Rouhani. « Towards Mobile Diminished Reality ». In : *Proceedings of the IEEE International Symposium for Mixed and Augmented Reality (ISMAR 2018)*. Oct. 2018. URL : https://www.ismar2018.org/papers/ismar2018%5C_demo%5C_24.html.

[Rad12] Richard J. Radke. *Computer Vision for Visual Effects*. English. Cambridge University Press, 2012. ISBN : 9780521766876, 0521766877.

[RHI] RHINOV. *Aménagement d'intérieur*. URL : https://www.rhinov.fr/comment-ca-marche.

[RMH10] Amandine Robin, Lionel Moisan et Sylvie Le Hegarat-Mascle. « An A-Contrario Approach for Subpixel Change Detection in Satellite Imagery ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.11 (nov. 2010), p. 1977–1993. DOI : 10.1109/tpami.2010.37. URL : https://doi.org/10.1109/tpami.2010.37.

[RT06] Guodong Rong et Tiow-Seng Tan. « Jump Flooding in GPU with Applications to Voronoi Diagram and Distance Transform ». In : *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*. I3D '06. Redwood City, California : ACM, 2006, p. 109–116. ISBN : 1-59593-295-X. DOI : 10.1145/1111411.1111431. URL : http://doi.acm.org/10.1145/1111411.1111431.

[RKB04] Carsten Rother, Vladimir Kolmogorov et Andrew Blake. « GrabCut : Interactive Foreground Extraction Using Iterated Graph Cuts ». In : *ACM Trans. Graph.* 23.3 (août 2004), p. 309–314. ISSN : 0730-0301.

[RP15] T. Ruzic et A. Pizurica. « Context-Aware Patch-Based Image Inpainting Using Markov Random Field Modeling ». In : *Image Processing, IEEE Transactions on* 24.1 (jan. 2015), p. 444–456. ISSN : 1057-7149. DOI : 10.1109/TIP.2014.2372479.

[STB18] S. H. Said, M. Tamaazousti et A. Bartoli. « Image-Based Models for Specularity Propagation in Diminished Reality ». In : *IEEE Transactions on Visualization and Computer Graphics* 24.7 (juil. 2018), p. 2140–2152. ISSN : 1077-2626. DOI : 10.1109/TVCG.2017.2705687.

[Sau18] Basile Sauvage. « Contributions à l'analyse et à la synthèse de l'apparence d'ob jets 3D numériques ». In : *Mémoire d'habilitation à diriger des recherches* (2018).

[Sch15] Jürgen Schmidhuber. « Deep learning in neural networks : An overview ». In : *Neural Networks* 61 (2015), p. 85–117. ISSN : 0893-6080. DOI : https://doi.org/10.1016/j.neunet.2014.09.003. URL : http://www.sciencedirect.com/science/article/pii/S0893608014002135.

[SH76] Michael Ian Shamos et Dan Hoey. « Geometric intersection problems ». In : *Foundations of Computer Science, 1976., 17th Annual Symposium on*. Oct. 1976, p. 208–215. DOI : 10.1109/SFCS.1976.16.

[SC02] J. Shen et T. Chan. « Mathematical Models for Local Nontexture Inpaintings ». In : *SIAM Journal on Applied Mathematics* 62.3 (2002), p. 1019–1043. DOI : 10.1137/S0036139900368844. eprint : https://doi.org/10.1137/S0036139900368844. URL : https://doi.org/10.1137/S0036139900368844.

[SKC03] J. Shen, S. Kang et T. Chan. « Euler's Elastica and Curvature-Based Inpainting ». In : *SIAM Journal on Applied Mathematics* 63.2 (2003), p. 564–592. DOI : 10.1137/S0036139901390088. eprint : https://doi.org/10.1137/S0036139901390088. URL : https://doi.org/10.1137/S0036139901390088.

[She68] Donald Shepard. « A Two-dimensional Interpolation Function for Irregularly-spaced Data ». In : *Proceedings of the 1968 23rd ACM National Conference*. ACM '68. New York, NY, USA : ACM, 1968, p. 517–524. DOI : 10.1145/800186.810616. URL : http://doi.acm.org/10.1145/800186.810616.

[Sil+12] Nathan Silberman, Derek Hoiem, Pushmeet Kohli et Rob Fergus. « Indoor Segmentation and Support Inference from RGBD Images ». In : *Proceedings of the Europeran Conference on Computer Vision (ECCV)*. 2012. DOI : 10.1007/978-3-642-33715-4_54. URL : https://doi.org/10.1007/978-3-642-33715-4_54.

[Sil15] Sanni Siltanen. « Diminished reality for augmented reality interior design ». English. In : *The Visual Computer* (2015), p. 1–16. ISSN : 0178-2789. DOI : 10.1007/s00371-015-1174-z. URL : http://dx.doi.org/10.1007/s00371-015-1174-z.

[Sin+08] Sudipta N. Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala et Marc Pollefeys. « Interactive 3D Architectural Modeling from Unordered Photo Collections ». In : *ACM Trans. Graph.* 27.5 (déc. 2008), 159 :1–159 :10. ISSN : 0730-0301.

[SSS06] Noah Snavely, Steven M. Seitz et Richard Szeliski. « Photo tourism : exploring photo collections in 3D ». In : *ACM Transactions on Graphics* 25.3 (juil. 2006), p. 835–846. ISSN : 07300301. DOI : 10.1145/1141911.1141964.

[Son+18] Yuhang Song *et al.*. « Contextual-based Image Inpainting : Infer, Match, and Translate ». In : *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018. DOI : 10.1007/978-3-030-01216-8_1.

[Str] Structure. *The Structure Sensor*. http://structure.io.

[Sun+05] Jian Sun, Lu Yuan, Jiaya Jia et Heung-Yeung Shum. « Image Completion with Structure Propagation ». In : *ACM Trans. Graph.* 24.3 (juil. 2005), p. 861–868. ISSN : 0730-0301. DOI : 10.1145/1073204.1073274. URL : http://doi.acm.org/10.1145/1073204.1073274.

[Sb85] Satoshi Suzuki et KeiichiA be. « Topological structural analysis of digitized binary images by border following ». In : *Computer Vision, Graphics, and Image Processing* 30.1 (1985), p. 32–46. ISSN : 0734-189X. DOI : https://doi.org/10.1016/0734-189X(85)90016-7. URL : http://www.sciencedirect.com/science/article/pii/0734189X85900167.

[TS10] Kosuke Takeda et Ryuuki Sakamoto. « Diminished Reality for Landscape Video Sequences with Homographies ». English. In : *Knowledge-Based and Intelligent Information and Engineering Systems*. Sous la dir. de Rossitza Setchi, Ivan Jordanov, RobertJ. Howlett et LakhmiC. Jain. T. 6278. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, p. 501–508. ISBN : 978-3-642-15392-1. DOI : 10.1007/978-3-642-15393-8_56. URL : http://dx.doi.org/10.1007/978-3-642-15393-8_56.

[TBU00] P. Thevenaz, T. Blu et M. Unser. « Interpolation revisited [medical images application] ». In : *IEEE Transactions on Medical Imaging* 19.7 (juil. 2000), p. 739–758. ISSN : 02780062. DOI : 10.1109/42.875199. URL : http://doi.org/10.1109/42.875199.

[TTL09] J. Y. Tou, Y. H. Tay et P. Y. Lau. « A Comparative Study for Texture Classification Techniques on Wood Species Recognition Problem ». In : *Proceedings of the Fifth International Conference on Natural Computation*. T. 5. Août 2009, p. 8–12. DOI : 10.1109/ICNC.2009.594.

[Tri+00] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley et Andrew W. Fitzgibbon. « Bundle Adjustment — A Modern Synthesis ». In : *Vision Algorithms : Theory and Practice*. Sous la dir. de Bill Triggs, Andrew Zisserman et Richard Szeliski. Berlin, Heidelberg : Springer Berlin Heidelberg, 2000, p. 298–372. ISBN : 978-3-540-44480-0.

[TD05] D. Tschumperlé et R. Deriche. « Vector-valued image regularization with PDEs : a common framework for different applications ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.4 (avr. 2005), p. 506–517. ISSN : 0162-8828. DOI : 10.1109/TPAMI.2005.87.

[Tsc12] David Tschumperlé. « The CImg Library ». In : *IPOL 2012 Meeting on Image Processing Libraries*. Cachan, France, juin 2012, 4 pp. URL : https://hal.archives-ouvertes.fr/hal-00927458.

[Tsc16] David Tschumperlé. *G'MIC - GREYC's Magic for Image Computing : A Full-Featured Open-Source Framework for Image Processing*. 2016. URL : https://gmic.eu/.

[TJ93] Mihran Tuceryan et Anil K. Jain. « Handbook of Pattern Recognition &Amp; Computer Vision ». In : sous la dir. de C. H. Chen, L. F. Pau et P. S. P. Wang. River Edge, NJ, USA : World Scientific Publishing Co., Inc., 1993. Chap. Texture Analysis, p. 235–276. ISBN : 981-02-1136-8. URL : http://dl.acm.org/citation.cfm?id=178866.178899.

[TPM06] Oncel Tuzel, Fatih Porikli et Peter Meer. « Region Covariance : A Fast Descriptor for Detection and Classification ». In : *Proceedings of the European Computer Vision Conference (ECCV 2006)*. Sous la dir. d'Aleš Leonardis, Horst Bischof et Axel Pinz. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 589–600. ISBN : 978-3-540-33835-2.

[Ven+13] Kartik Venkataraman *et al.*. « PiCam : An Ultra-thin High Performance Monolithic Camera Array ». In : *ACM Trans. Graph.* 32.6 (nov. 2013), 166 :1–166 :13. ISSN : 0730-0301. DOI : 10.1145/2508363.2508390. URL : http://doi.acm.org/10.1145/2508363.2508390.

[WMG14] Michael Waechter, Nils Moehrle et Michael Goesele. « Let there be color ! Large-scale texturing of 3D reconstructions ». In : *Proceedings of the 2014 European Conference on Computer Vision (ECCV 2014)*. T. 8693 LNCS. PART 5. 2014, p. 836–850. ISBN : 978-3-319-10601-4. DOI : 10.1007/978-3-319-10602-1_54.

[Wan+14] Jing Wang, Ke Lu, Daru Pan, Ning He et Bing-kun Bao. « Robust object removal with an exemplar-based image inpainting approach ». In : *Neurocomputing* 123 (2014). Contains Special issue articles : Advances in Pattern Recognition Applications and Methods, p. 150–155. ISSN : 0925-2312. DOI : https://doi.org/10.1016/j.neucom.2013.06.022. URL : http://www.sciencedirect.com/science/article/pii/S0925231213006334.

[Wan+18] Yi Wang, Xin Tao, Xiaojuan Qi, Xiaoyong Shen et Jiaya Jia. « Image Inpainting via Generative Multi-column Convolutional Neural Networks ». In : *arXiv preprint arXiv :1810.08771* (2018). URL : https://arxiv.org/abs/1801.07892.

[Wan+04] Zhou Wang, A. C. Bovik, H. R. Sheikh et E. P. Simoncelli. « Image quality assessment : from error visibility to structural similarity ». In : *IEEE Transactions on Image Processing* 13.4 (avr. 2004), p. 600–612. ISSN : 1057-7149. DOI : 10.1109/TIP.2003.819861.

[Way] Wayfair. *The Wayfair App*. URL : https://www.wayfair.com/the-wayfair-app.

[WSI07] Y. Wexler, E. Shechtman et M. Irani. « Space-Time Completion of Video ». In : *IEEE Transactions on Pattern Analysis & Machine Intelligence* 29.3 (mar. 2007), p. 463–476. ISSN : 0162-8828. DOI : 10.1109/TPAMI.2007.60. URL : doi.ieeecomputersociety.org/10.1109/TPAMI.2007.60.

[WCM15] Yu-Shiang Wong, Hung-Kuo Chu et Niloy J. Mitra. « SmartAnnotator An Interactive Tool for Annotating Indoor RGBD Images ». In : *Computer Graphics Forum* 34.2 (2015), p. 447–457. ISSN : 1467-8659. DOI : 10.1111/cgf.12574. URL : http://dx.doi.org/10.1111/cgf.12574.

[WR08] Jiying Wu et Qiuqi Ruan. « A novel hybrid image inpainting model ». In : *Proceedings of the International Conference on Audio Language and Image Processing (ICALIP 2008)*. Juil. 2008, p. 138–142. DOI : 10.1109/ICALIP.2008.4589952.

[Xia+11] Chunxia Xiao, Meng Liu, Nie Yongwei et Zhao Dong. « Fast Exact Nearest Patch Matching for Patch-Based Image Editing and Processing ». In : *IEEE Transactions on Visualization and Computer Graphics* 17.8 (août 2011), p. 1122–1134. ISSN : 1077-2626. DOI : 10.1109/TVCG.2010.226.

[Yan+16] S. Yang, J. Liu, S. Song, M. Li et Z. Quo. « Structure-guided image completion via regularity statistics ». In : *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2016, p. 1711–1715. DOI : 10.1109/ICASSP.2016.7471969.

[Yu+18] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu et Thomas S Huang. « Generative Image Inpainting with Contextual Attention ». In : *arXiv preprint arXiv :1801.07892* (2018).

[YSK11] Cem Yuksel, Scott Schaefer et John Keyser. « Parameterization and applications of Catmull–Rom curves ». In : *Computer-Aided Design* 43.7 (2011). The 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling, p. 747–755. ISSN : 0010-4485. DOI : https://doi.org/10.1016/j.cad.2010.08.008. URL : http://www.sciencedirect.com/science/article/pii/S0010448510001533.

[ZCC16] Edward Zhang, Michael F. Cohen et Brian Curless. « Emptying, Refurnishing, and Relighting Indoor Spaces ». In : *ACM Trans. Graph.* 35.6 (nov. 2016), 174 :1–174 :14. ISSN : 0730-0301. DOI : 10.1145/2980179.2982432.

[Zha+18] Haoran Zhang, Zhenzhen Hu, Changzhi Luo, Wangmeng Zuo et Meng Wang. « Semantic Image Inpainting with Progressive Generative Networks ». In : *Proceedings of the 26th ACM International Conference on Multimedia*. MM '18. Seoul, Republic of Korea : ACM, 2018, p. 1939–1947. ISBN : 978-1-4503-5665-7. DOI : 10.1145/3240508.3240625. URL : http://doi.acm.org/10.1145/3240508.3240625.

[Zha+11] G. Zhao, G. Wu, Y. Liu et J. Chen. « Texture Classification Based on Completed Modeling of Local Binary Pattern ». In : *Proceedings of the International Conference on Computational and Information Sciences*. Oct. 2011, p. 268–271. DOI : 10.1109/ICCIS.2011.271.

[Zhi18] Jixiang Cheng Zhidan Li. « Exemplar-based image inpainting using structural feature offsets statistics ». In : *Proceedings of the Tenth International Conference on Digital Image Processing*. T. 10806. SPIE, 2018, DOI : 10.1117/12.2502934. URL : https://doi.org/10.1117/12.2502934.

[Zok+03] Siavash Zokai, Julien Esteve, Yakup Genc et Nassir Navab. « Multiview Paraperspective Projection Model for Diminished Reality ». In : *Proceedings of the 2Nd IEEE/ACM International Symposium on Mixed and Augmented Reality*. ISMAR '03. Washington, DC, USA : IEEE Computer Society, 2003, p. 217–. ISBN : 0-7695-2006-5. URL : http://dl.acm.org/citation.cfm?id=946248.946801.