جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITE SIDI MOHAMED BEN ABDALLAH DE FES
كلية العلوم ظهر المهراز فاس
FACULTE DES SCIENCES DHAR EL MEHRAZ FES



N° d'Ordre : **27/2021**

Formation Doctorale : Sciences et Technologies de l'Information et de la Communication (STIC)

Discipline : Informatique

Spécialité : Informatique

Laboratoire : LISAC

THESE DE DOCTORAT

Présentée par

Laaroussi Saadeddine

Contributions aux algorithmes de mosaïques d'images : Cas des scènes contenant des objets dynamiques

Soutenue le 10/06/2021 devant le jury composé de :

Pr. TAIRI Hamid	***Faculté des sciences Dhar El Mahraz - Fès***	***Président***
Pr. SBAI EL Hassan	***Ecole Supérieure de Technologie – Meknès***	***Rapporteur***
Pr. ROUKHE Ahmed	***Faculté des sciences – Meknès***	***Rapporteur***
Pr. EL ABDERRAHMANI Abdellatif	***Faculté des sciences Dhar El Mahraz - Fès***	***Rapporteur***
Pr. EL AKKAD Nabil	***Ecole Nationale des Sciences Appliqués - Fès***	**Examinateur**
Pr. SATORI Hassan	***Faculté des sciences Dhar El Mahraz - Fès***	**Examinateur**
Pr. SABRI My Abdelouahed	***Faculté des sciences Dhar El Mahraz - Fès***	***Examinateur***
Pr. HALLI Akram	***Faculté des Sciences Juridiques, Economiques et Sociales - Meknès***	**Co-Directeur de thèse**
Pr. SATORI Khalid	***Faculté des sciences Dhar El Mahraz - Fès***	**Directeur de thèse**

Année universitaire : 2020-2021

Remerciements

C’est avec le plus grand honneur que je réserve cette page pour exprimer toute ma gratitude et remerciements aux personnes qui ont aidé à l’accomplissement et la réalisation de ce travail.

Tout premièrement, je souhaite exprimer mes sincères remerciements au professeur, Monsieur **Khalid Satori**, mon directeur de thèse, qui m’as initié au domaine de la recherche et m’as encadré tout au long de ce projet. Je le remercie également pour sa disponibilité permanente, son expertise scientifique et pédagogique, ses directives prospectives, ainsi que tous les échanges constructives qu’on a souvent eu.

Je souhaite également exprimer toute ma reconnaissance à Monsieur **Akram HALLI**, professeur de la Faculté des sciences juridiques, économiques et sociales de Meknès, pour son aide précieuse, ses encouragements, sa gentillesse et sa grande disponibilité. Merci de m’avoir toujours encouragé, de m’avoir fait confiance et de m’avoir toujours aidé.

Je tiens aussi à exprimer particulièrement mes remerciements à Monsieur **Aziz BAATAOUI**, professeur de la faculté sciences et techniques d’Errachidia, pour tous ses conseils précieux, sa patience, sa disponibilité permanente et tous les efforts qu’il a effectué pour que cette recherche soit d’une bonne qualité.

Chers encadrants, grâce à vous j’ai réussi à développer mes compétences scientifiques, j’ai appris énormément de vous et ce fut un grand honneur d’être sous votre encadrement. Cette recherche fut une expérience inoubliable et je vous exprime mon plus grand respect et mes remerciements les plus profonds.

Je souhaite également remercier les honorables membres du jury : J’exprime toute ma gratitude à Monsieur **Hamid TAIRI**, professeur de la faculté des sciences Dhar el Mahraz de Fès, d’avoir accepter de présider ce jury.

Je suis très reconnaissant des professeurs, Monsieur **SBAI EL Hassan**, professeur de l’école Supérieure de Technologie de Meknès, Monsieur **ROUKHE Ahmed**, professeur de la faculté des sciences de Meknès et Monsieur **EL ABDERRAHMANI Abdellatif**, de la faculté des sciences Dhar el Mahraz de Fès, d’avoir accepter d’évaluer ce mémoire en tant que rapporteurs et de leurs diverses contributions au manuscrit final.

Je remercie également les membres examinateurs de ce jury : Monsieur **EL AKKAD Nabil**, de l'Ecole Nationale des Sciences Appliqués de Fès, Monsieur **SATORI Hassan**, de la faculté des sciences Dhar el Mahraz de Fès et Monsieur **SABRI My Abdelouahed**, de la faculté des sciences Dhar el Mahraz de Fès, pour l'effort et l'intérêt qu'ils ont accordé à ce travail.

Finalement, je remercie de tout mon cœur, mon père, ma mère et ma sœur Atika, qui m'ont soutenu et encouragé tout au long de cette expérience. Je ne saurai jamais vous remercier pour tout ce que vous m'avez apporté.

Résumé

Une mosaïque d'images est définie par une succession d'algorithmes utilisés sur un ensemble d'entrée, qui peut être soit des images soit une vidéo de la même scène, pour former une seule image contenant toutes les informations de la scène de l'ensemble d'entrées avec un meilleur champ de vision. Le but de ce sujet de recherche est de trouver un moyen d'améliorer le résultat de la mosaïque obtenue dans des scènes contenant des objets dynamiques. En effet, les objets dynamiques sont définis comme tout changement entre les différentes images d'entrées qui peuvent générer des erreurs, que ce soit des éléments dans l'avant-plan comme des éléments vivants ou machines, ou des éléments dans l'arrière-plan qui changent d'une manière discrète telles que des particules. Ainsi, sans traitement de ces objets dynamiques, des erreurs comme le "ghosting", effets parallaxes et dédoublement d'objets apparaissent et créent de mauvais résultats. Ces erreurs peuvent être néfastes dans certains domaines d'applications où une imprécision ou duplication d'informations n'est pas souhaité.

Dans cette thèse, nous avons tout d'abord présenté les généralités et les étapes pour la création d'une mosaïque d'images de haute qualité. Ensuite, nous avons expliqué les différentes méthodes de la littérature qui permettent de palier aux différents problèmes de mosaïques d'images dynamiques, ainsi que les procédures pour l'évaluation de ces dernières.

Dans l'objectif de réaliser cette recherche, nous avons proposé trois nouvelles méthodes. Les deux premières méthodes consistent à chercher une coupure optimale qui va séparer la zone de chevauchement des images sans créer de discontinuités. La première méthode développée est basée sur l'utilisation des "inliers" et "ouliers" pour créer des régions avec le détecteur de Canny pour calculer le chemin optimale d'une manière dynamique, ce chemin-ci est calculé d'une manière pour qu'il passe par des régions sûres ("inliers") et évite les autres régions ("outliers"). Tandis que la deuxième méthode consiste à utiliser l'algorithme A* sur une carte d'énergie optimisée créée avec un mouvement fractionnaire Brownien modifié avec la fonction "Zero Normalized Cross Correlation" au lieu d'une fonction de bruit aléatoire pour trouver le meilleur chemin. La troisième méthode est une méthode basée sur la reconstruction des régions. Cette dernière utilise un système de recommandation basé sur les objets et le filtrage collaboratif sur toutes les images simultanément. Les images d'entrées seront considérées comme des utilisateurs, les pixels et leur valeurs comme les objets et leur scores, et la mosaïque finale comme un nouvel utilisateur au système qui va être généré par une recommandation.

Mots-clés : Mosaïques d'images dynamiques, Recalage d'images, "Blending" d'images, Algorithme A*, Système de recommandation.

Abstract

An image mosaic is a succession of algorithms used on an entry set, which can either be a set of images or a video of a scene, to create a single image that contains all the information of the scene with a wider field of view. The main purpose of this research is to improve the mosaic result obtained from a scene with dynamic objects. In fact, dynamic objects are defined as any possible change that can be observed between the entry images that can create errors either in the foreground like living objects or machines, or in the background like particles. As such, without treating these dynamic objects, errors like ghosting (remnant of dynamic objects), parallax effects (errors in stitching) and object duplication appear and cause poor results. Furthermore, these errors can be harmful in some application domains like the medical domain or the motion tracking domain where an imprecision or duplication of information is not wanted.

In this thesis,first of all, we introduced the subject of image mosaicking in a global way by discussing the different concepts and the steps to create a high quality mosaic image. Afterwards, we explored the different methods in the literature that treat the issues of dynamic image mosaicking, as well as the different approaches to evaluate dynamic mosaics.

In order to achieve the main objective of this research, we proposed three novel methods. The first two methods consist on finding a way to compute an optimal seamline that will separate the overlap areas of the images without creating any discontinuities. The first proposed method uses inliers and outliers to create regions with a Canny edge detector in order to compute the optimal path in a dynamic manner. This path is obtained by passing through common regions (inliers) while avoiding the other regions (outliers). The second method combines between the pathfinding algorithm A* and an optimized energy map created by a modified fractional Brownian motion calculated with a Zero Normalized Cross Correlation (ZNCC) to find the optimal seamline instead of a random noise function. The third approach is a region reconstruction method. In fact, an object centered recommender system with a collaborative filter was used on all the images to create the final mosaic. The input images are considered as users, the pixels and their values as objects and their scores, and the mosaic is introduced to the system as a new user.

Keywords : Mosaicking, Dynamic Mosaicking, Image registration, Image blending, A* algorithmn, Recommender system.

Table des matières

Remerciements

Résumé

Abstract

Liste des figures **v**

Liste des tableaux **viii**

Liste des abréviations **x**

Introduction générale **1**

Problématique 2
Motivations 3
Contributions 4
Organisation du mémoire 5

1 Généralités sur les mosaïques d'images dynamiques **6**

1.1 Introduction 7
1.2 Généralités sur les mosaïques d'images dynamiques 7
1.2.1 Mosaïques d'images dynamiques 7
1.2.2 Domaines d'applications 8
1.2.3 Etapes pour la création d'une mosaïque d'images dynamiques 10
1.3 Recalage d'images 10
1.3.1 Hiérarchie de transformation 11
1.3.2 Estimation de l'homographie 13
1.3.2.1 Cas de deux images 13
1.3.2.2 Cas de plusieurs images 14
1.3.3 Catégories des méthodes de recalage d'images 14
1.3.4 Recalage d'images dans les mosaïques d'images 15
1.3.4.1 Détecteur de points d'intérêts 15
1.3.4.2 Mise en correspondance et RANSAC 18

1.4 “Blending” d’images . . . 21
1.4.1 Méthodes de “blending” d’images . . . 21
1.4.1.1 “Alpha blending” . . . 21
1.4.1.2 “Feathering” . . . 22
1.4.1.3 “Multi-band blending” . . . 22
1.4.1.4 “Gradient domain blending” . . . 22
1.5 Prétraitement et Posttraitement . . . 24
1.5.1 Prétraitement . . . 24
1.5.1.1 Correction d’exposition . . . 25
1.5.2 Posttraitement (Redressage d’images) . . . 25
1.6 Conclusion . . . 27

2 Traitement des objets dynamiques dans les mosaïques d’images 28
2.1 Introduction . . . 29
2.2 Artefacts . . . 29
2.3 Gestion des objets dynamiques . . . 30
2.3.1 Méthodes basées sur le calcul d’une “seamline” optimale . . . 32
2.3.1.1 Méthode de Mills et Dudek . . . 33
2.3.1.2 Méthode de Zeng et al. . . . 34
2.3.1.3 Méthode de Yao et al. . . . 36
2.3.1.4 Méthode de Li et al. . . . 37
2.3.2 Méthodes basées sur la reconstruction de régions . . . 39
2.3.2.1 Méthode d’Uyttendaele et al. . . . 39
2.3.2.2 Méthode de Bartoli et al. . . . 40
2.3.3 Comparaison des méthodes . . . 42
2.4 Evaluation des mosaïques . . . 42
2.4.1 Procédure d’évaluation . . . 43
2.4.2 Mesures quantitatives . . . 44
2.4.2.1 “Structure Similarity Index Measure” (SSIM) . . . 44
2.4.2.2 “Multi-Scale Structure Similarity Index Measure” (MS-SSIM) . . . 45
2.4.2.3 “Spectral Angle Mapper” (SAM) . . . 45
2.4.2.4 “Intensity Magnitude Ratio” (IMR) . . . 46
2.4.3 Evaluation des mosaïques d’images dynamiques . . . 46
2.5 Conclusion . . . 48

3 Création de mosaïques d’images dynamiques basées sur un détecteur de contours pour calculer une “seamline” optimale. 49
3.1 Introduction . . . 50
3.2 Utilisation d’un détecteur de contours pour calculer la “seamline” optimale . . . 52
3.2.1 Prétraitement : Transfert de couleurs . . . 52

3.2.2 Recalage d'images . . . 54
3.2.3 Recherche de la "seamline" avec un détecteur de contours . . . 54
3.2.3.1 Régions Canny . . . 54
3.2.3.2 Carte d'énergie . . . 56
3.2.3.3 Calcul de la "seamline" optimale . . . 57
3.2.4 "Pyramid blending" . . . 59
3.3 Résultats expérimentaux . . . 60
3.3.1 Données synthétiques . . . 60
3.3.2 Données réelles . . . 64
3.3.3 Analyses des mesures obtenues . . . 70
3.4 Conclusion . . . 70

4 Mosaïque d'images basée sur le mouvement fractionnaire Brownien et l'algorithme A*. 71
4.1 Introduction . . . 72
4.2 Carte d'énergie fBm . . . 73
4.2.1 Recalage d'images . . . 75
4.2.2 Création de la carte d'énergie . . . 75
4.2.2.1 Mouvement fractionnaire Brownien . . . 75
4.2.2.2 "Zero Normalized Cross Correlation (ZNCC)" . . . 76
4.2.2.3 Génération de la carte d'énergie par FBm modifié et ZNCC . . . 77
4.3 Résultat mosaïque avec l'algorithme A* . . . 79
4.3.1 Calcul de la "seamline" A* . . . 79
4.3.2 "Pyramid blending" . . . 81
4.4 Analyses et résultats expérimentaux . . . 83
4.4.1 Analyses sur les paramètres . . . 83
4.4.1.1 Analyses sur le nombre de distributions . . . 84
4.4.1.2 Analyses sur la fréquence initiale . . . 86
4.4.1.3 Tests sur le chemin de la "seamline" . . . 87
4.4.2 Résultats expérimentaux . . . 88
4.4.3 Analyses des mesures obtenues . . . 94
4.5 Conclusion . . . 94

5 Mosaïque d'images basée sur un système de recommandation. 95
5.1 Introduction . . . 96
5.2 Segmentation des images avec un "clustering" . . . 97
5.2.1 Recalage d'images . . . 97
5.2.2 "Clustering" K-means . . . 99
5.2.3 Catégorisation des pixels . . . 100
5.3 Création d'une mosaïque avec un système de recommandation . . . 103

5.3.1 Création des profils des pixels . . . 103
5.3.2 Recommandation de la mosaïque . . . 104
5.4 Résultats expérimentaux . . . 106
5.4.1 Résultats . . . 109
5.4.2 Analyses des mesures obtenues . . . 113
5.5 Conclusion . . . 114

Conclusion générale et perspectives **115**

A Outils mathématiques **118**
A.1 Introduction . . . 119
A.2 “Least square method” . . . 119
A.3 Décomposition en valeurs singulières (SVD) . . . 120
A.4 Différentes interpolations . . . 121
A.5 Conclusion . . . 122

B Utilisation du flux optique dense pour créer une mesure quantitative pour l’évaluation d’une mosaïque **123**
B.1 Introduction . . . 124
B.2 Flux optique dense . . . 124
B.3 Résultats expérimentaux . . . 125
B.4 Conclusion . . . 128

Bibliographie **129**

Publications scientifiques **136**

Liste des figures

1.1 Exemple d'une mosaïque d'images dynamiques près de mausolée Mohammed V à Rabat. 8
1.2 Exemples de domaines d'applications, (a) Mosaïque d'une thyroïde sans tissue par Shapiro [11] (b) Mosaïque créée avec des drones par Shen et al. [14] (c) Motion Panorama par Bartoli et al.[17] . . . 9
1.3 Séries d'algorithmes générique pour la création d'une mosaïque d'images dynamiques . . 10
1.4 Différents effets qu'une matrice de transformation peut effectuer sur l'image (a). Image capturée à Bab Mansour à Meknès. . . . 12
1.5 Pyramide de taille $n_{pyr} = 5$ avec Pyr_G la pyramide Gaussian et Pyr_{DoG} la pyramide Laplacian/Difference of Gaussian . . . 16
1.6 Algorithme SIFT . . . 18
1.7 Algorithme RANSAC . . . 20
1.8 Exemple de "multi-band blending" (d) Résultat sans blending (e) Résultat avec blending. . 23
1.9 Mosaïque (a) Avant le redressage (b) Après le redressage avec un angle de 2.9° calculé avec Hough. . . . 27

2.1 Artefact, (a) Effet parallaxe, (b) Ghosting. . . . 30
2.2 Exemple de carte d'énergie de Mills et Dudek [54], (a) Différence d'intensité (Equation 2.2), (b) Différence de gradient (Equation 2.3), (c) Carte d'énergie (Equation 2.1) . . . 34
2.3 Exemple de carte d'énergie Zeng et al. [55], les éléments seuillés sont en vert. . . . 35
2.4 Procédure de choix du prochain pixel pour la méthode CSDP. . . . 35
2.5 Procédure de "blending" de la méthode de Yao et al. [37] . . . 37
2.6 Exemple de carte de distance Li et al. [56], (a) Images d'entrées, (b)Différence au carré des canaux CIELab D_c, (b) Différence des histogrammes de gradient HoG D_h, (c) Carte de distance D(Equation 2.11) . . . 38
2.7 Exemple de création de graphe RoD d'Uyttendaele et al. [62] . . . 40
2.8 Exemple de la méthode Bartoli et al. [17], (a) Images d'entrées, (b) Carte de probabilité p, (c) Carte de probabilité après filtre Gaussian, (d) Avant-plan de l'image d'entrée du milieu obtenue avec la carte de probabilité après filtre Gaussian (e) Carte de probabilité après seuillage (f) Avant-plan de l'image d'entrée du milieu après seuillage. . 42
2.9 Procédure de calcul du score de qualité d'une image avec une méthode NR-IQA . . . 47

3.1 Procédure pour la création d'une mosaïque avec la méthode basée sur un détecteur de contours. [71]. 51
3.2 Application du transfert de couleur sur l'ensemble d'images Leuven de la base de données "Oxford standard data set". 53
3.3 Procédure de créations des régions Canny. 55
3.4 Procédure d'obtention de la carte d'énergie finale. 57
3.5 Choix des points adjacents pour le calcul de la "seamline". 58
3.6 Calcul de la "seamline" et mosaïque finale, (a) Tableau *sign* appliqué sur $\delta^{Im'}$ avec les valeurs positives en bleu et négatives en jaune, (b) "Seamline" choisie en vert parmi les différentes "seamlines" calculées en rouge, (c) Mosaïque finale après le "blending". . . . 58
3.7 Algorithme de la méthode développée basée sur le détecteur de contours de Canny pour le calcul d'une "seamline" optimale et la création d'une mosaïque. 59
3.8 Paires d'images de la base de données DAVIS (a) Motocross-jump (b) Horsejump-low (c) Bus (d) Mallard-fly (e) Boxing-fisheye. 61
3.9 "Seamlines" des images de la base de données DAVIS (a) Motocross-jump (b) Horsejump-low (c) Bus (d) Mallard-fly (e) Boxing-fisheye, (1) Mills (2) CSDP (3) Notre méthode. 62
3.10 Mosaïques des images de la figure 3.8 (a) Motocross-jump (b) Horsejump-low (c) Bus (d) Mallard-fly (e) Boxing-fisheye, (1) Mills (2) CSDP (3) Notre méthode. 63
3.11 Temps d'exécution des méthodes Mills, CSDP et notre méthode pour les images de la figure 3.8. 64
3.12 Paires d'images capturées (a) People (b) Valley (c) Cars (d) Plaza (e) Door. 65
3.13 Séries des données réelles de la figure 3.12 (a) Image de référence (b) Seconde image avant le prétraitement (c) Résultat après le prétraitement. 66
3.14 "Seamlines" des images des données réelles de la figure 3.12 : (a) People (b) Valley (c) Cars (d) Plaza (e) Door, (1) Mills (2) CSDP (3) Notre Méthode. 67
3.15 Mosaïques des images de la figure 3.12 (a) People (b) Valley (c) Cars (d) Plaza (e) Door, (1) Mills (2) CSDP (3) Notre Méthode. 68
3.16 Temps d'exécution des méthodes Mills, CSDP et la méthode développée pour les images de la figure 3.12. 69

4.1 Procédure pour la création d'une mosaïque avec la méthode basée sur le mouvement factionnaire Brownien et l'algorithme A*. [78]. 74
4.2 Exemple de mise à jour de la fréquence à chaque distribution sur un pixel avec un rayon initial $r = 1$. Le pixel choisi est le pixel sombre au milieu. 77
4.3 Création de la carte d'énergie (a) I_1 et I_2 les images d'entrées, (b) Trois distributions calculées avec $ZNCC_{fBm}$ en utilisant la zone de chevauchement des images I_1 et I_2. De gauche à droite, première distribution avec un rayon initial de $f_1 = r = 1$, la seconde $f_2 = 2$, et la dernière distribution $f_3 = 4$ (c) Carte d'énergie E avec une persistance de $p = 0.75$. 79

4.4 Calcul de la “seamline” avec A* (a) Carte d’obstacle extraite de la carte d’énergie E de la figure 4.3.c, (b) “Seamline” A* qui commence par le point qui a la valeur la plus petite de la première ligne (c) “Seamline” A* qui commence par le premier point qui n’est pas un obstacle. 81
4.5 Résultat mosaïque après le “pyramid blending” (a) avec bordures (b) sans bordures. . . . 82
4.6 Algorithme de la méthode développée basée sur l’utilisation d’un fBm modifié avec ZNCC 82
4.7 Images utilisées pour l’analyse (a) Première série de la base DAVIS (bmx-trees) (b) Deuxième série Plaza. 84
4.8 Cartes d’énergie (a, c) et zones de chevauchements des résultats (b, d) dans différents nombre de distributions créés à partir des images de la figure 4.7. 84
4.9 Temps d’exécution pour la création de la carte d’énergie et le calcul de la “seamline”, ainsi que leur somme pour (a) Figure 4.7.a et (b) Figure 4.7.b sur différentes distributions. 85
4.10 Cartes d’énergie et les “seamlines” de la série d’images Figure 4.7.a sous différentes fréquences et distributions. 86
4.11 Temps d’éxecution de la création de la carte d’énergie et du calcul de la “seamline” en secondes pour la série de la figure 4.7.a avec différentes fréquences maximales en variant la fréquence initiale et le nombre de distributions. 87
4.12 Zones de chevauchements des mosaïques résultantes de la figure 4.7.b où : (a) Aucune modification, (b) Bruit, (c) Sous-exposition, (d) Sur-exposition, (e) Bruit + Sous-exposition, (f) Buit + sur-exposition. Avec un temps d’exécution pour le calcul de la “seamline” de (a) $2,409s$ (b) $2,438s$ (c) $2,501s$ (d) $2,156s$ (e) $2,393s$ (f) $2,085s$. 88
4.13 Séries utilisées pour la partie expérimentale, (a) Bmx-trees, (b) Scooter-gray, (c) Valley, (d) Lindy-hop and (e) Bridge 89
4.14 Résultats mosaïques pour les séquences : (a) Bmx-trees (b) Scooter-gray ; où les logiciels et méthodes sont (1) Logiciel Hugin, (2) Logiciel PTGui, (3) méthode CSDP, (4) méthode Graphcut, (5) méthode Inlier/Outlier, (6) Méthode développée. 90
4.15 Résultats mosaïques pour les séquences : (a) Valley ; où les logiciels et méthodes sont (1) Logiciel Hugin, (2) Logiciel PTGui, (3) méthode CSDP, (4) méthode Graphcut, (5) méthode Inlier/Outlier, (6) Méthode développée. 91
4.16 Résultats mosaïques pour les séquences : (a) Lindy-hop (Fig. 12.d), (b) Bridge (Fig. 12.e) ; où les logiciels et méthodes sont (1) Logiciel Hugin, (2) Logiciel PTGui, (3) méthode CSDP, (4) méthode Graphcut, (5) méthode Inlier/Outlier, (6) Méthode développée. 92
4.17 Temps d’exécution des différentes approches pour toutes les séquences en secondes. . . . 93

5.1 Etapes pour la recommandation d’une mosaïque avec plusieurs images d’entrées. 98
5.2 Images d’entrées après le recalage d’images. 99
5.3 Classes Cl obtenues des images de la figure 5.2 avec l’algorithme k-means, $k = 3$. 100
5.4 Zones similaires O_q obtenues de chaque “cluster” Cl_q de la Figure 5.3 et les zones non similaires détectées $O^{"}$ qui peuvent causer des artefacts. 101

5.5 Segmentation finale (a) Catégories obtenues en combinant toutes les zones de la figure 5.4, (b) Zones de chevauchements (c) L'ensemble C qui contient la combinaison de toutes les catégories et les zones de chevauchements (Figure 5.5.a et 5.5.b). 102
5.6 Résultat mosaïque des images de la figure 5.2. 105
5.7 Algorithme de la méthode développée basée sur un système de recommandation. 106
5.8 Série Car. 107
5.9 Série Cat. 107
5.10 Série Shibuya. 108
5.11 Série Vault. 108
5.12 Série High-jump. 108
5.13 Résultats obtenus de la série Car (Figure 5.8), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode. 109
5.14 Résultats obtenus de la série Cat (Figure 5.9), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode. 110
5.15 Résultats obtenus de la série Shibuya (Figure 5.10), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode. 111
5.16 Résultats obtenus de la série Vault (Figure 5.11), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode. 112
5.17 Résultats obtenus de la série High-jump (Figure 5.12), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode. 112
5.18 Temps d'exécution des différentes approches. 113

B.1 Exemple de création de zones avec le flux optique dense, (a) Images d'entrées, (b) Flux optique dense des images, (c) Extractions de zones. 126
B.2 Images utilisées pour la comparaison des mesures tirées de la base de données DAVIS avec (a) bear, (b) breakdance, (c) dog-agility, (d) drift-straight, (e) kite-surf et (1) la zone de chevauchement des images, (2) résultat d'une méthode statique (3) mauvais résultat d'une méthode basée sur une "seamline" (4) bon résultat d'une méthode basée une "seamline". 127

Liste des tableaux

3.1 Valeurs PSNR obtenues entre l'image de référence (Figure 3.2.a) et les images avant le transfert de couleurs (Figure 3.2.b) et après le transfert de couleurs (Figure 3.2.c) 54
3.2 Valeurs SSIM, SAM et IMR pour les mosaïques créées à partir de la figure 3.8 64
3.3 Valeurs PSNR obtenues entre les paires des images réelles de la figure 3.12) 66

3.4 Valeurs SSIM, SAM et IMR pour les mosaïques créées à partir de la figure 3.12 69

4.1 Valeurs de SSIM, MS_SSIM, SSIM_Seamline pour les méthodes CSDP, Graphcut, Inlier/Outlier et notre méthode pour les mosaïques obtenues à partir des images de la figure 4.13. (Valeurs optimaux en gras) 93

5.1 Valeurs SSIM et MS-SSIM de : RoD, CSDP, Graphcut, fBm_A* et notre méthode pour les mosaïques obtenues des figures 5.8, 5.9, 5.10, 5.11, 5.12. (Valeurs optimaux en gras) . . 110

B.1 Valeurs des mesures d'évaluation obtenues avec les images de la figure B.2 avec SSIM avec un bloc de 16 et de 256 et la mesure de SSIM développée avec le flux optique dense. 128

Liste des abbréviations

BRISQUE	Blind/Referenceless Image Spatial Quality Evaluator
CSDP	Combined SIFT with dynamic programming
DoG	Difference of Gaussian
FBm	Fractional Brownian motion
FFT	Fast Fourier Transform
FLANN	Fast Library for Approximate Nearest Neighbors
HoG	Histogram of Gradient
HSV	Hue, Saturation, Value
IMR	Intensity Magnitude Ratio
LoG	Laplacian of Gaussian
LSH	Locality Sensitive Hashing
LSM	Least square method
MLESAC	Maximum Likelihood Estimate SAmple Consensus
MS-SSIM	Multi-Scale Structure Similarity Index Measure
MSAC	M-estimator SAmple and Consensus
NIQE	Naturalness Image Quality Evaluator
NR-IQA	No-Reference Image Quality Assessment
PIQE	Perception based Image Quality Evaluator
RANSAC	RANdom SAmple Consensus
RGB	Red, Green, Blue
RoD	Regions of Difference
SAM	Spectral Angle Mapper
SIFT	Scale-invariant feature transform
SSIM	Structure Similarity Index Measure
SURF	Speeded-Up Robust Features
SVD	Singular Value Decomposition
SVR	Support Vector Regression
SURF	Speeded-Up Robust Features
UAV	Unmanned Aerial Vehicles
ZNCC	Zero Normalized Cross Correlation

Introduction générale

La vision par ordinateur est un sujet vaste qui permet aux machines de décrire, d'analyser et d'interpréter les images et vidéos. Plusieurs domaines utilisent la vision par ordinateur pour réaliser certaines tâches comme la reconstruction d'objets et de scènes, la reconnaissance de formes, la détection et le suivi de mouvement etc... Parmi ces différentes tâches, on trouve la création de mosaïques d'images.

Une mosaïque d'images est une grande image permettant d'afficher une scène complète avec un meilleur champ de vision et une bonne qualité. Cette mosaïque est créée à partir de plusieurs images d'entrées d'une scène. Cette définition est très similaire à celle d'un panorama mais une différence entre les deux existe. En effet, un panorama est créé avec des images capturées par une caméra qui reste physiquement statique avec la direction d'acquisition d'images qui change d'une image à une autre, cependant pour une mosaïque, la caméra n'a pas besoin d'être physiquement statique. Ainsi, la création de panoramas est un des domaines d'applications des mosaïques d'images. De plus, les mosaïques sont souvent utilisées comme un outil dans plusieurs domaines de recherches [1], tels que les chirurgies minimalement invasives [2], les drones ("Unmanned Aerial Vehicles" (UAV)) [3], la création d'images satellites [4] etc...

En principe, une mosaïque est définie comme un ensemble d'algorithmes où différentes images d'entrées sont combinées pour obtenir une représentation graphique complète d'une scène en évitant la création de discontinuités. En général, deux séries principales d'algorithmes sont utilisées, la première série étant le recalage d'images qui projette les données d'entrées dans le même système de coordonnées grâce aux différentes méthodes directes ou basées sur les caractéristiques d'images etc... et qui détecte les points correspondants dans les images grâce aux méthodes de mise en correspondance. Et la deuxième série, le "blending" d'images où plusieurs méthodes, qui sont basées sur le lissage de la zone de transition ou sur la recherche de la coupe optimale, permettent d'assembler les images pour obtenir un effet "seamless". Cependant, la création d'une mosaïque n'est pas limitée à ce processus. Effectivement, plusieurs algorithmes supplémentaires peuvent être ajoutés à cette procédure que ce soit en prétraitement ou en posttraitement. Par exemple, pour faciliter la création de la mosaïque, un prétraitement est effectué sur les images afin d'enlever le bruit ou d'égaliser les couleurs entre les images avec une correction d'exposure ou une égalisation d'histogramme [5] etc..., ou comme autre exemple, un posttraitement est appliqué sur la mosaïque résultante pour enlever l'effet

d'horizon courbé, un effet souvent obtenu quand plusieurs images sont utilisées, avec un algorithme de redressage d'images ("Image Straightening") [6] etc... De plus, l'utilisation d'un prétraitement ou d'un posttraitement est dépendant du domaine de recherche exploré. Entre autres, pour la création de mosaïques sous-marine [7], c'est préférable d'effectuer un prétraitement pour améliorer les images d'entrées alors que pour les mosaïques ultrasons ou satellitaires, enlever les granulations du bruit Speckle est recommandé etc...

Problématique

Durant ces dernières années, malgré l'avancement de la technologie, il n'existe actuellement pas de matériels ou de logiciels proposant une solution pour créer une mosaïque d'une manière optimale, même les méthodes existantes dans ce domaine de recherche ne fonctionnent pas dans certains cas. En effet, des erreurs apparaissent fréquemment, en conséquence, l'utilisation de logiciels tiers pour créer les mosaïques manuellement reste toujours populaires.

Concernant les appareils récents, le système optique des caméras traditionnelles qui suivent un modèle sténopé (Pinhole Camera) et qui utilisent une projection perspective n'est pas suffisant pour obtenir un grand champ de vision dans les images capturées. Même avec l'apparition des caméras omnidirectionnelles [8], qui introduisent une solution aux autres modèles de projections en augmentant le champ de vision à 360°, certaines erreurs restent fréquentes. En effet, un logiciel tiers est souvent utilisé, il est inclus soit dans la caméra, soit hors caméra. Ce dernier est utilisé pour définir une ligne nommée "stitch line" qui permet de rassembler l'image, ce qui est souvent fait manuellement.

Pour la création d'une mosaïque, plusieurs logiciels existent dans la littérature, AutoStitch [9] parmi les premiers, ce logiciel est performant mais comme la majorité des logiciels il est soumis à certaines conditions pour obtenir un résultat sans erreurs. Dans les récentes années, certains logiciels comme Hugin et PTGui (logiciel professionnel) qui sont à jour, essayent d'introduire des solutions mais dans certaines situations les erreurs restent toujours visible.

Ainsi, une mosaïque d'une méthode qui donne un bon résultat, peut être définie comme une mosaïque ne contenant aucune discontinuité. Cependant, il faut suivre des règles strictes pour obtenir ce résultat dans toutes les situations :

1) Les images doivent être capturées par une caméra qui tourne autour de son centre optique, souvent effectuée avec une caméra fixe ou un trépied ;
2) La scène doit être statique. En effet, les différentes images capturées doivent plus ou moins avoir la même zone de chevauchement sans la présence de différences flagrantes ;
3) Les images doivent être capturées au même moment et avoir les mêmes intensités.

En pratique, ces conditions sont difficiles à réaliser et limitent l'utilisateur dans certains domaines

d'applications. De plus, la création de mosaïques devient plus complexe lorsque les images d'entrées utilisées sont capturées à partir d'une scène dynamique (objets en mouvement, environnement etc...). Effectivement, plusieurs problèmes peuvent apparaître, et ces complications peuvent être situationnelles ou techniques. Parmi les erreurs les plus fréquentes, on retrouve :

- Le "*ghosting*", un effet qui apparaît lorsque la scène n'est pas statique et contient des objets dynamiques. Ces derniers peuvent aussi bien faire partie de l'avant-plan (foreground), en tant qu'objet qui a changé de position d'une image à une autre, que faire partie de l'arrière-plan (background), en tant que texture qui a légèrement changé entre les images. En outre, le ghosting est considéré comme le reste des objets dynamiques dont des parties incomplètes sont présentes dans la mosaïque finale suite à une mauvaise fusion des images d'entrées.
- Les *effets parallaxes*, des distorsions qui apparaissent quand la scène capturée n'est pas planaire. Véritablement, cet effet apparaît fréquemment. D'autant plus si plusieurs images d'entrées sont utilisées pour la création de la mosaïque. Souvent, il est dû à la mauvaise projection des images dans le même repère.

Les problèmes cités auparavant (ci-dessus) ne sont pas limitées aux objets dynamiques mais aussi aux algorithmes utilisés dans les phases de recalage et de "blending" d'images. En plus, le risque d'erreurs augmente quand le nombre d'images d'entrées est élevé, là où une accumulation d'erreurs se produit plus facilement.

Motivations

Plusieurs travaux existent déjà dans le domaine des mosaïques d'images mais la majorité traitent le cas des scènes statiques. Les motivations et buts de ce sujet de recherche qu'on souhaite réaliser sont différents puisqu'il concerne le thème de mosaïques d'images **dynamiques**.

Tout d'abord, avant d'entamer nos objectifs, il faut savoir que les méthodes statiques qui créent des mosaïques à partir de scènes statiques tendent à se concentrer sur la manière d'améliorer le recalage d'images afin d'obtenir un bon assemblage des images durant la phase du "blending". Cependant, pour les méthodes dynamiques, contrairement aux méthodes statiques, le recalage d'images n'as pas un rôle aussi important. En effet, les algorithmes de traitement des différences entre les zones de chevauchement permettent de démasquer les différentes erreurs issues de la projection des images et les traitent. Ainsi, les problèmes qu'on souhaite régler et les objectifs qu'on désire accomplir sont :

- Rechercher de nouveaux moyens pour la gestion et détection des objets dynamiques présents dans les différentes images afin d'éliminer les artefacts mentionnés précédemment ("Ghosting" et effets parallaxes), ainsi que d'autres erreurs non souhaitées comme par exemple la duplication d'objets, où ce dernier est un effet qui survient dans la mosaïque résultante affichant une duplication d'informations par rapport aux images d'entrées.

- Réaliser des algorithmes rapides en améliorant la façon de traiter les zones homogènes dans les images. Ces zones sont souvent la cause des ralentissements dans plusieurs méthodes.
- Créer des méthodes robustes permettant de gérer plusieurs images en même temps au lieu de deux images à la fois comme ce qu'on peut trouver actuellement dans la littérature pour espérer une future création en temps réel.
- Produire des mosaïques de qualité même si les images d'entrées ont été capturées avec des restrictions, que ce soit des images capturées d'une scène sans trépied, à des temps différents avec une différence d'exposure ou avec des caméras différentes etc...

Contributions

Plusieurs contributions ont été effectuées afin de réaliser nos objectifs :

- Premièrement, nous avons commencé par la recherche d'un moyen pour créer une "seamline" optimale. Cette dernière est une ligne qui sépare la zone de chevauchement de deux images, ainsi dans l'assemblage, les pixels à gauche de la ligne sont pris de la première image et ceux à droite sont pris de la seconde image. Par conséquent, pour que la "seamline" soit optimale c'est impératif qu'elle ne traverse aucune partie dynamique. Ainsi, nous avons développé une nouvelle méthode qui permet de surligner les régions similaires et les régions non similaires dans la zone de chevauchement des images en utilisant les "inliers" et "outliers" obtenus dans la phase du recalage pour contrôler la "seamline". En effet, cette dernière a été calculé de sorte qu'elle passe par les zones similaires et évite les zones non similaires pour ne créer aucun artefact. En parallèle, afin de faciliter le calcul de la recherche du chemin optimal, nous avons utilisé un transfert de couleur pour corriger la différence d'exposition qui existe entre les images, ainsi qu'une égalisation d'histogramme pour améliorer la détection des objets dynamiques. A noter que même si les images sont capturées par la même caméra, une exposition existera toujours suite aux différentes positions des sources de lumières.
- Deuxièmement, nous avons développé un nouveau moyen pour calculer la "seamline" en utilisant l'apprentissage automatique. En effet, une nouvelle carte d'énergie a été créée pour détecter les pixels susceptibles de générer des artefacts en utilisant le mouvement fractionnaire Brownien ("fractional Brownian motion" (fBm)) avec une fonction de similarité prédéterminée au lieu d'une fonction de bruit. Cette fonction est la "Zero Normalized Cross Correlation" (ZNCC). Cette action a permis d'assurer l'unicité de la valeur des pixels de la carte d'énergie générée même dans les zones homogènes. Ainsi, en combinant la nouvelle carte d'énergie avec l'algorithme d'apprentissage automatique A*, le meilleur chemin pour générer une "seamline" optimale est obtenu. De plus, suite aux difficultés de comparer les mosaïques d'images dynamiques nous avons aussi créé une nouvelle méthode de comparaison pour calculer une mesure quantitative en utilisant le flux optique dense.

- Finalement, nous avons recherché un moyen d'utiliser toutes les images d'entrées en même temps. En général, quand plusieurs images sont utilisées pour créer une mosaïque, on utilise seulement deux images à la fois. En effet, les deux premières images génèrent une mosaïque temporaire, qui elle-même est utilisée pour générer la prochaine mosaïque temporaire avec l'image suivante, jusqu'au moment où il ne reste plus d'images d'entrées. Ainsi, nous avons introduit un système de recommandation hybride basé sur les objets avec un filtre collaboratif ("item-to-item collaborative filtering recommender system"). En effet, en considérant les images comme des utilisateurs et les pixels comme des objets, c'est possible d'utiliser un système de recommandation où la mosaïque est considérée comme un nouveau utilisateur auquel le système va lui recommander les pixels par rapport aux scores des images d'entrées. Par conséquent, les pixels des images ont été classées en différentes catégories après la phase du recalage avec "k-means" pour calculer un score spécifique à chaque pixel pour une recommandation précise.

Organisation du mémoire

Ce mémoire est organisé de la façon suivante :

- Dans le chapitre 1, nous présenterons les généralités sur les mosaïques d'images dynamiques. Une définition détaillée sera présentée, ainsi que les différents domaines d'applications des mosaïques. Ensuite, les étapes de création des mosaïques d'images dynamiques seront établies avec un focus sur le recalage d'images, le blending d'images, le prétraitement et le posttraitement.
- Dans le chapitre 2, nous définirons un objet dynamique et les différents artefacts qui apparaissent lorsque les objets dynamiques sont mal gérés. Ensuite, les différentes méthodes de la littérature vont être explorées. Et finalement, la procédure d'évaluation des mosaïques d'images dynamiques sera présentée.
- Dans les chapitres 3, 4 et 5, nous présenterons les méthodes développées pour le traitement des objets dynamiques dans le domaine des mosaïques d'images dynamiques. Les méthodes des chapitres 3 et 4 sont des méthodes basées sur le calcul d'une "seamline" optimale. Tout d'abord, dans le chapitre 3, la méthode que nous présenterons est basée sur la création des régions avec le détecteur de contours Canny. Puis, dans le chapitre 4, la méthode présentée est basée sur un mouvement fractionnaire Brownien combiné avec l'algorithme d'apprentissage automatique A*. Tandis que la méthode du chapitre 5 est une méthode basée sur la reconstruction de régions où on utilisera un système de recommandation.

Chapitre **1**

Généralités sur les mosaïques d'images dynamiques

1.1 Introduction . . . 7
1.2 Généralités sur les mosaïques d'images dynamiques . . . 7
1.2.1 Mosaïques d'images dynamiques . . . 7
1.2.2 Domaines d'applications . . . 8
1.2.3 Etapes pour la création d'une mosaïque d'images dynamiques . . . 10
1.3 Recalage d'images . . . 10
1.3.1 Hiérarchie de transformation . . . 11
1.3.2 Estimation de l'homographie . . . 13
1.3.2.1 Cas de deux images . . . 13
1.3.2.2 Cas de plusieurs images . . . 14
1.3.3 Catégories des méthodes de recalage d'images . . . 14
1.3.4 Recalage d'images dans les mosaïques d'images . . . 15
1.3.4.1 Détecteur de points d'intérêts . . . 15
1.3.4.2 Mise en correspondance et RANSAC . . . 18
1.4 "Blending" d'images . . . 21
1.4.1 Méthodes de "blending" d'images . . . 21
1.4.1.1 "Alpha blending" . . . 21
1.4.1.2 "Feathering" . . . 22
1.4.1.3 "Multi-band blending" . . . 22
1.4.1.4 "Gradient domain blending" . . . 22
1.5 Prétraitement et Posttraitement . . . 24
1.5.1 Prétraitement . . . 24
1.5.1.1 Correction d'exposition . . . 25
1.5.2 Posttraitement (Redressage d'images) . . . 25
1.6 Conclusion . . . 27

1.1 Introduction

Ce premier chapitre présente les généralités sur les mosaïques d'images. Ainsi, plusieurs points vont être traités notamment des définitions de bases, les domaines d'applications et les différentes étapes pour la création d'une mosaïque d'images d'une scène contenant des objets dynamiques. En plus, plusieurs algorithmes et méthodes qui sont spécifiques à chaque étape dans la création de mosaïques seront définies en détails.

1.2 Généralités sur les mosaïques d'images dynamiques

1.2.1 Mosaïques d'images dynamiques

Une mosaïque d'image est une image synthétisée créée à partir de plusieurs images ou une vidéo de la même scène avec une série d'algorithmes dont le but est l'obtention d'un résultat final qui est une réprésentation graphique de toute la scène sans créer aucune discontinuité.

Deux domaines de recherche peuvent être distingués pour les mosaïques d'images [1]. Le premier étant pour les images et les vidéos qui possèdent une zone de chevauchement, c'est à dire que les images contiennent une zone commune. Dans ce cas, on parle d'"Image Stitching" ou de "Photo Stitching" (Assemblage d'images). Cependant pour le second domaine, les images n'ont pas de zones de chevauchement. Par conséquence, dans cette situation on parle d'"Image Compositing"(Composition d'images) [10]. Les deux domaines ont des procédures différentes. En effet, le premier domaine essaye de rester fidèle un maximum aux informations obtenues dans les images d'entrées tandis que le deuxième crée de nouveaux pixels à partir des données d'entrées. Mais au final, ils possèdent le même but dans la création d'une image représentant la même scène avec un meilleur champ de vision.

Il existe deux types de mosaïques, les mosaïques d'images statiques et les mosaïques d'images dynamiques. La première est créée à partir d'une succession d'images d'une scène statique, par conséquence, les zones de chevauchements des données d'entrées sont quasi similaires. Ainsi, un pixel qui se trouve dans une certaine position avec une certaine intensité aura la même position et intensité dans toutes les images. Cependant, pour les mosaïques d'images dynamiques, les zones communes des images se diffèrent d'une image à une autre et ces différences sont *des objets dynamiques*. En outre, les objets dynamiques sont définis comme les dissemblances qui existent dans un certain nombre d'images mais qui n'existent pas dans d'autres. Ces incohérences peuvent être des objets physiques qui ont changé d'espaces d'une image à une autre, comme ils peuvent être des textures qui ont été modifié suite à des forces externes ou en raison d'un changement d'exposure à cause du système optique de la caméra lors de l'acquisition des images ou à cause d'un changement des sources de lumières.

Ainsi, ce sujet de recherche va se concentrer sur le domaine d'assemblage des images pour traiter le cas des images d'une scène contenant des objets dynamiques. La figure 1.1 présente un exemple d'une mosaïque d'images dynamiques créée à partir de 10 images d'entrées capturées avec une caméra.



(a) Image d'entrées



(b) Résultat

FIGURE 1.1 – Exemple d'une mosaïque d'images dynamiques près de mausolée Mohammed V à Rabat.

1.2.2 Domaines d'applications

Les domaines d'applications des mosaïques d'images dynamiques sont vastes. En effet, ces derniers sont utilisés comme un outil dans plusieurs domaines pour atteindre un certain objectif. Cependant, on peut regrouper tous ces domaines d'applications dans le domaine de la vision par ordinateur et d'infographie. Parmi ces domaines on retrouve :

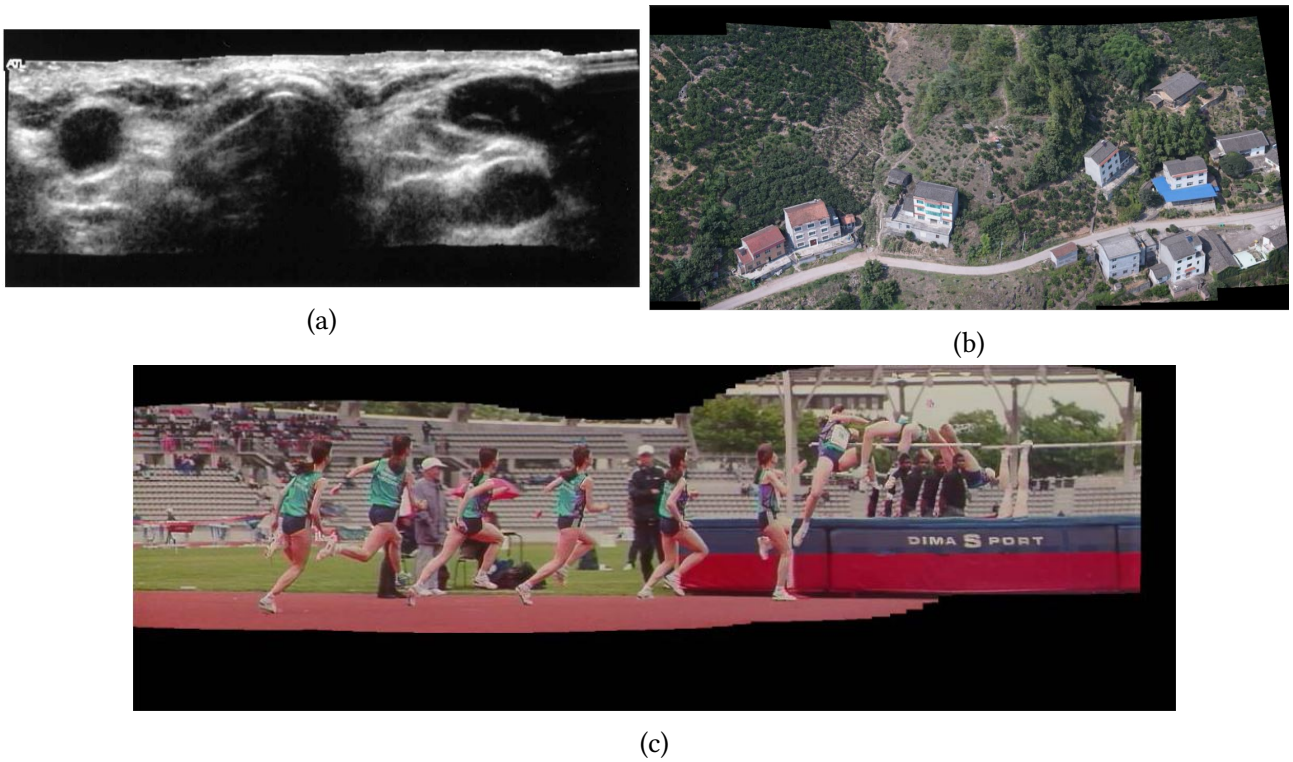
- **Domaine médical :** les mosaïques sont de plus en plus utilisées dans ce domaine, plusieurs applications existent notamment : (i) pour faciliter les diagnostiques en créant des mosaïques pour certaines maladies auto-immunitaires comme par exemple les maladies liées à la thyroïde [11] ou liées au changement de la masse musculaire [12]. (ii) améliorer la visibilité des docteurs lors des chirurgies minimalement invasive par exemple l'endomicroscopie [13] où la caméra scanne et crée une mosaïque en temps réel etc...
- **Domaine des robots et d'exploration :** dans ce domaine les robots sont utilisés pour explorer les zones dangereuses et inaccessibles pour les humains ou pour des missions de reconnaissances, parmi les robots les plus utilisés il y a les drones [14] ("Unmanned Aerial Vehicles" (UAV)). D'autres applications existent notamment dans le cas de la création des mosaïques sous-marines [15].
- **Domaine de détection d'objets et suivi de mouvement :** Vu que les mosaïques d'images dynamiques possèdent une étape de gestion des objets dynamiques, par conséquence, elles peuvent être utilisées dans la détection d'objets et le suivi de mouvement. Ainsi, plusieurs recherches utilisent les mosaïques pour : (i) La détection d'objets, par exemple pour l'identification d'objets dans des milieux sous-marins [16] qui est utile dans des recherches comme la géologie, la biologie ou l'excavation de bateaux. (ii) Le suivi de mouvement d'un objet dyna-

mique où une mosaïque (une motion panorama) contenant toutes les positions de cet objet à tous les instants est générée [17].

- **Domaine de l'ortho photographie :** Ce domaine basé sur l'imagerie satellitaire et l'imagerie aérienne utilise les mosaïques pour la création de grandes cartes avec une géo rectification et l'égalisation de la radiométrie. C'est l'un des domaines où les mosaïques sont les plus utilisées notamment pour la création des cartes de villes en combinant les images en passant par les routes tout en évitant les bâtiments et objets dynamiques [4] sans créer de discontinuités. Un exemple assez connu est un site web de la NASA [18] qui montre plusieurs mosaïques créées à partir de plusieurs images capturées par des satellites dans des scènes différentes.
- etc...

La figure 1.2 montre quelques exemples de mosaïques créées dans différents domaines.

Les mosaïques ne sont pas limitées aux domaines d'applications présentés dans cette sous-section. En effet, les domaines indiqués sont parmi les plus utilisés et représentent juste une partie des possibilités où les mosaïques peuvent être utilisées, d'autres domaines existent comme la reconstruction 3D [18, 19] ou la réalité augmenté [20, 21] etc...



(a) (b) (c)

FIGURE 1.2 – Exemples de domaines d'applications, (a) Mosaïque d'une thyroïde sans tissue par Shapiro [11] (b) Mosaïque créée avec des drones par Shen et al. [14] (c) Motion Panorama par Bartoli et al.[17]

1.2.3 Etapes pour la création d'une mosaïque d'images dynamiques

La création d'une mosaïque d'images dynamiques inclue une étape supplémentaire par rapport à la mosaïque d'une scène statique, c'est l'étape de la gestion des objets dynamiques. Ainsi, les étapes par ordre sont le recalage d'images, la gestion des objets dynamiques et le "blending" d'images, avec éventuellement une étape de prétraitement ou posttraitement. La figure 1.3 résume les différents étapes.

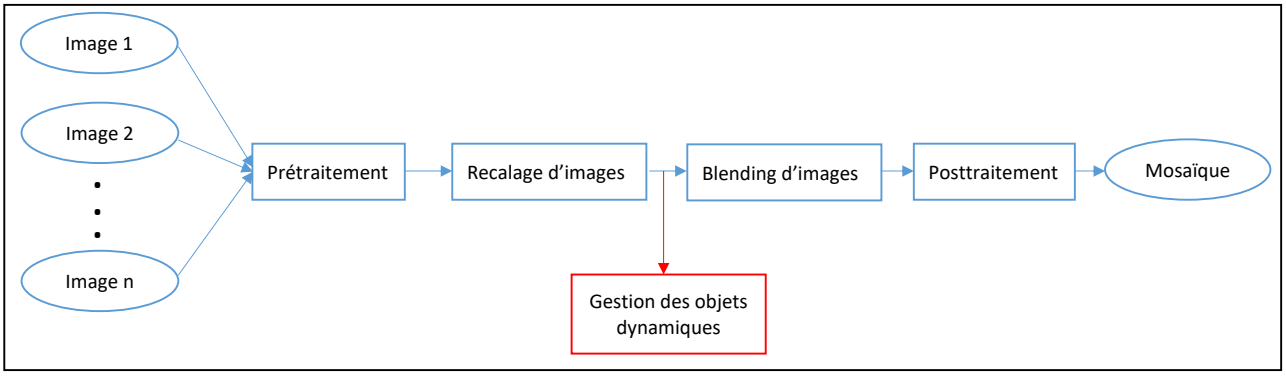


FIGURE 1.3 – Séries d'algorithmes générique pour la création d'une mosaïque d'images dynamiques

1.3 Recalage d'images

Le recalage d'images est la première étape dans la création d'une mosaïque. L'objectif du recalage d'images est de projeter toutes les images d'entrées dans le même système de coordonnées. En outre, une matrice de transformation H de taille $(3, 3)$ est calculée tel que chaque position d'un pixel P de l'image Im_1 peut être retrouvé dans l'image Im_2 avec cette matrice. L'équation 1.1 représente cette formulation.

$$P = T_H * P^{'} + R_H \Leftrightarrow P = H_{Im_2Im_1} \times P^{'} \tag{1.1}$$

Où :

- P et $P^{'}$ représentent les coordonnées homogènes du même pixel dans l'image Im_1 et Im_2 respectivement.
- T_H la matrice de translation et R_H la matrice de rotation.
- $H_{Im_2Im_1}$ la matrice de transformation projetant les pixels de l'image Im_1 au système de coordonnées de l'image Im_2.

Quand le produit est effectué entre la matrice de transformation et un point, le résultat ne donne pas forcément des coordonnées entières, mais des valeurs réelles. Vu qu'une image ne peut pas posséder des coordonnées dans le domaine réel, une interpolation doit être effectuée. Il existe différents types d'interpolations (Annexe A) qu'on utilise dans le recalage d'images :

- "*nearest*" : La valeur d'un pixel de coordonnées (x, y) est prise de la valeur du pixel avec les coordonnées les plus proche à (x, y).
- "*linear*" : La valeur d'un pixel de coordonnées (x, y) est calculée avec une fonction d'interpolation bilinéaire avec les deux pixels les plus proches de (x, y).
- "*cubic*" : La valeur d'un pixel de coordonnées (x, y) est calculée avec une fonction d'interpolation bicubique avec les quatre pixels les plus proches de (x, y).

1.3.1 Hiérarchie de transformation

Le calcul de la matrice de transformation est le point le plus important dans le recalage d'images puisqu'une bonne matrice minimisera les erreurs de projections dans la mosaïque finale. Il existe différents types de matrices de transformations qui peuvent être effectuées sur les images. Ces matrices sont aussi caractérisées par un degré de liberté. Effectivement, plus le degré de liberté est élevé, plus l'image qui va être projeté sera distordue par rapport à son état initial. En plus, ces matrices sont soumises à une hiérarchie tel que les matrices avec un degré bas sont inclus dans les matrices avec un degré plus élevé que leur propre degré. Les matrices de transformations selon le degré de liberté sont listées ci-dessous :

- **Transformation de translation :** avec 2 degré de liberté, elle traite la translation par rapport à deux axes, la matrice s'écrit de la façon suivante :

$$H_{translation} = \begin{bmatrix} 1 & 0 & t_a \\ 0 & 1 & t_o \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

  Où t_a et t_o représentent la translation effectuée par rapport à l'axe des abscisses et des ordonnées respectivement.

- **Transformation euclidienne :** avec 3 degré de liberté, en plus de la translation cette matrice traite aussi la rotation, elle est formulée par l'expression :

$$H_{euclidienne} = \begin{bmatrix} cos(\theta) & -sin(\theta) & t_a \\ sin(\theta) & cos(\theta) & t_o \\ 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

  où θ est l'angle de rotation.

- **Transformation de similarité :** avec 4 degré de liberté, cette matrice ajoute l'échelle à la rotation et à la translation, elle est calculée avec la matrice suivante :

$$H_{similarit} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \times H_{euclidienne} \tag{1.4}$$

  Où s représente le facteur d'échelle.

- **Transformation affine :** avec 6 degré de liberté, elle contient les éléments de transformations des matrices précédentes et rajoute le traitement du ratio projection ("aspect ratio") et de la transvection géométrique, elle est formulée par l'expression ci-dessous :

$$H_{affine} = \begin{bmatrix} r & 0 & 0 \\ 0 & \frac{1}{r} & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & t_1 & 0 \\ t_2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times H_{similarit} \tag{1.5}$$

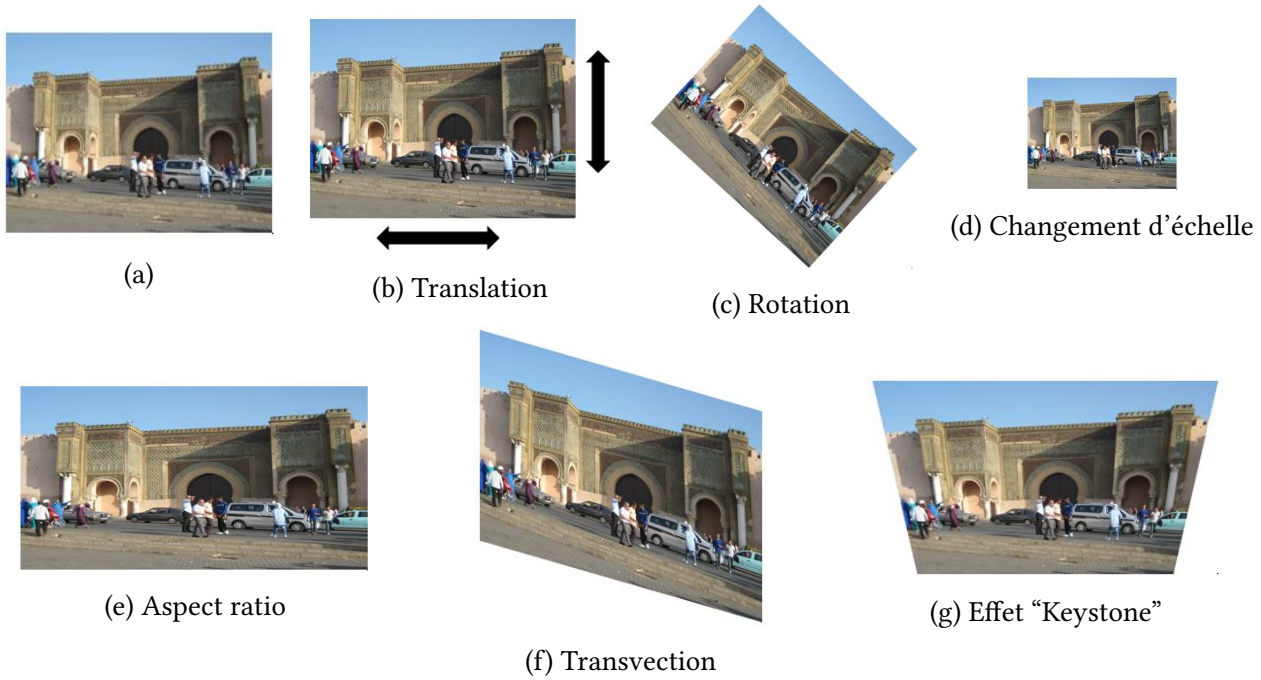
Avec r est la valeur de l'aspect ratio et t_1 et t_2 les facteurs de transvection.

- **Transformation projective :** aussi appelée transformation perspective, avec 8 degré de liberté elle permet de traiter tout type de modification entre deux images. En effet, en plus de la translation, la rotation, l'échelle, l'aspect ratio et la transvection, cette matrice traite aussi l'effet "keystone". Elle est aussi appelée l'homographie et elle est définie par l'équation :

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \tag{1.6}$$

avec h_7 et h_8 qui sont responsable de l'effet keystone.

La figure 1.4 présente les différents effets qui peuvent être obtenus avec une matrice de transformation. En outre, une combinaison de tous ces effets est obtenue avec l'homographie.



(a) (b) Translation (c) Rotation (d) Changement d'échelle (e) Aspect ratio (f) Transvection (g) Effet "Keystone"

FIGURE 1.4 – Différents effets qu'une matrice de transformation peut effectuer sur l'image (a). Image capturée à Bab Mansour à Meknès.

1.3.2 Estimation de l'homographie

Suite à la nature de capture aléatoire des images dans le domaine des mosaïques d'images dynamiques, la transformation projective est celle qui est utilisée pour projeter les images dans le même repère. Il existe deux cas différents pour le calcul de l'homographie :

1.3.2.1 Cas de deux images

Soit deux images successives Im_1 et Im_2 avec une zone commune. Pour calculer l'homographie, il faut 4 paires de points minimum avec leurs coordonnées homogènes connues dans les deux images pour obtenir une solution. En outre, le point P_1 de l'image Im_1 possède une position correspondante $P_1^{'}$ connue dans l'image Im_2 et ainsi de suite.

Puisque l'homographie est unique entre tous les points des deux images et en utilisant l'équation 1.1, on obtient 8 équations avec 8 inconnus, après simplification un système d'équations linéaires 1.7 est obtenue.

$$\underbrace{\begin{bmatrix} P_{1x} & P_{1y} & 1 & 0 & 0 & 0 & -P_{1x}\times P_{1x}^{'} & -P_{1y}\times P_{1x}^{'} & -P_{1x}^{'} \\ 0 & 0 & 0 & P_{1x} & P_{1y} & 1 & -P_{1x}\times P_{1y}^{'} & -P_{1y}\times P_{1y}^{'} & -P_{1y}^{'} \\ P_{2x} & P_{2y} & 1 & 0 & 0 & 0 & -P_{2x}\times P_{2x}^{'} & -P_{2y}\times P_{2x}^{'} & -P_{2x}^{'} \\ 0 & 0 & 0 & P_{2x} & P_{2y} & 1 & -P_{2x}\times P_{2y}^{'} & -P_{2y}\times P_{2y}^{'} & -P_{2y}^{'} \\ P_{3x} & P_{3y} & 1 & 0 & 0 & 0 & -P_{3x}\times P_{3x}^{'} & -P_{3y}\times P_{3x}^{'} & -P_{3x}^{'} \\ 0 & 0 & 0 & P_{1x} & P_{1y} & 1 & -P_{3x}\times P_{3y}^{'} & -P_{3y}\times P_{3y}^{'} & -P_{3y}^{'} \\ P_{4x} & P_{4y} & 1 & 0 & 0 & 0 & -P_{4x}\times P_{4x}^{'} & -P_{4y}\times P_{4x}^{'} & -P_{4x}^{'} \\ 0 & 0 & 0 & P_{4x} & P_{4y} & 1 & -P_{4x}\times P_{4y}^{'} & -P_{4y}\times P_{4y}^{'} & -P_{4y}^{'} \end{bmatrix}}_{\text{Matrice A}} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{bmatrix} = 0 \tag{1.7}$$

Avec P_{ix}, P_{iy} les coordonnées des points P_i et $P_{ix}^{'}$, $P_{iy}^{'}$ les coordonnées des points $P_i^{'}$ avec i= 1..4.

La solution du système d'équations linéaires présentée dans l'équation 1.7 peut être obtenue en appliquant une décomposition en valeurs singulières ("Singular Value Decomposition (SVD)") (Annexe A) sur la matrice A. En effet, on obtiendra : $A_{(8,9)} = U_{(8,8)} \times \sum_{(8,9)} \times V_{(9,9)}^{T}$ avec la solution étant la dernière colonne de la matrice V^T. Par conséquence, la matrice H peut être calculée par l'équation ci-dessous :

$$H = \begin{bmatrix} \frac{V_{1,9}^T}{V_{9,9}^T} & \frac{V_{2,9}^T}{V_{9,9}^T} & \frac{V_{3,9}^T}{V_{9,9}^T} \\ \frac{V_{4,9}^T}{V_{9,9}^T} & \frac{V_{5,9}^T}{V_{9,9}^T} & \frac{V_{6,9}^T}{V_{9,9}^T} \\ \frac{V_{8,9}^T}{V_{9,9}^T} & \frac{V_{8,9}^T}{V_{9,9}^T} & 1 \end{bmatrix} \tag{1.8}$$

1.3.2.2 Cas de plusieurs images

Quand n images sont projetées dans le même système de coordonnées, tout d'abord, une image de référence Im_r est choisie, ensuite les homographies entre toutes les images successives sont calculées. Ainsi, pour calculer la matrice de transformation projective entre l'image Im_r et une image Im_i, toutes les homographies calculées entre ces deux images sont multipliées. Cette expression est formulée par l'équation 1.9 où les pixels de l'image Im_i sont projetés dans le système de coordonnées de l'image Im_r dans le cas où $i < r$.

$$H_{Im_r,Im_i} = H_{Im_r,Im_{r-1}} \times H_{Im_{r-1},Im_{r-2}} \times ... \times H_{Im_{i+1},Im_i} \tag{1.9}$$

Avec r l'indice de l'image de référence.

Par conséquent, après la multiplication d'homographie et l'obtention de la matrice de transformation, il faut vérifier les bordures des images et les ajuster pour ne pas obtenir des pixels de l'image projetée hors des bordures de l'image. En outre, cet effet peut résulter en un manque d'information dans les images à cause de l'obtention de pixels projetés qui sont négatifs. Ainsi, afin d'éviter ce genre d'erreur, il faut calculer la nouvelle position pour seulement les quatre coins de l'image Im_i avec la matrice H_{Im_r,Im_i} avant la projection de l'image entière. Si les pixels des bordures sont négatifs, il faut mettre à jour H_{Im_r,Im_i} en la multipliant par une matrice de translation (Equation 1.2) contenant les valeurs permettant de remettre les pixels à l'intérieur des bordures. A noter que c'est préférable d'effectuer cette mise à jour après chaque multiplication pour éviter une accumulation d'erreurs.

1.3.3 Catégories des méthodes de recalage d'images

Plusieurs méthodes existent pour effectuer le recalage d'images [23] dont l'objectif est la projection des images d'entrées dans le même repère en calculant la matrice de transformation. Ainsi, il faut trouver les points correspondants dans les images d'entrées, ce qui est accomplissable de plusieurs manières, mais en général les méthodes peuvent être assemblées en différentes catégories, parmi les plus populaires :

- *Méthodes manuelles :* Ces méthodes consistent à choisir les points de contrôles manuellement. Même si ces méthodes sont plus aptes à créer des erreurs, elles permettent à l'utilisateur de choisir les points similaires entre deux images pour calculer la transformation géométrique selon ses objectifs. Le recalage manuel est simple à utiliser et à implémenter, en plus, ce qui rend cette catégorie toujours populaire malgré l'existence de méthodes automatiques c'est la possibilité de se concentrer sur une section d'une image particulière pour étudier plus précisément cette région d'intérêt et ainsi ignorer les autres régions, ce qui n'est pas faisable avec les autres méthodes.

- *Méthodes directes :* Ces méthodes utilisent une mesure de similarité et comparent directement les images soit pixel par pixel (recalage local) ou région par région (recalage global) pour trouver les pixels similaires et calculer la transformation géométrique. Ainsi, plusieurs types de méthodes existent dans cette catégorie, on retrouve des méthodes de corrélation, des méthodes de minimisation d'intensité etc...
- *Méthodes basées sur les caractéristiques de l'image :* Ces méthodes consistent à chercher les points d'intérêts qui caractérisent chaque image, ces points tendent à être des contours, des coins, des régions fermés, des intersections, des jonctions, des variations en intensité etc... Puis les points de chaque image sont comparés avec les points des autres images pour ensuite estimer le modèle de transformation. C'est une méthode de recalage local qui se base sur la comparaison pixel par pixel.

1.3.4 Recalage d'images dans les mosaïques d'images

La majorité des méthodes de création de mosaïques d'images tendent à utiliser des méthodes locales basées sur les caractéristiques de l'image. En outre, l'homographie est calculée en trois étapes : l'utilisation d'un détecteur de points d'intérêts, la mise en correspondance des paires de points d'intérêts correctes et l'estimation de l'homographie.

1.3.4.1 Détecteur de points d'intérêts

Les détecteurs de points d'intérêts ont deux rôles importants. Le premier est de trouver tous les pixels qui sont covariants aux différentes transformations qu'ils peuvent subir et le deuxième est de décrire ces pixels d'une façon unique afin que les pixels similaires aient la même description.

Dans le domaine des mosaïques, il existe plusieurs moyens pour détecter les points d'intérêts. Les détecteurs "Scale-Invariant Feature Transform" (SIFT) [24] et "Speeded-Up Robust Features" (SURF) [25] sont les plus courants dans les méthodes de création de mosaïques notamment les méthodes dynamiques [26, 27, 28]. D'autres détecteurs, moins courants comme le détecteur A-KAZE [29] ou Harris [30] sont aussi utilisés pour créer des mosaïques [17, 31, 32].

Dans cette thèse, nous avons décidé d'utiliser le détecteur SIFT pour trois raisons principales. Premièrement, ce détecteur de points d'intérêts est robuste et efficace d'après plusieurs études comparatives [33, 34]. En plus, il est aussi performant dans la création de mosaïques sous différentes conditions lumineuses [35], ce qui peut réduire l'apparition d'erreurs. Deuxièmement, l'algorithme SIFT, non seulement effectue la détection de points d'intérêts, mais aussi la description ce qui n'est pas le cas pour d'autres algorithmes où il faut ajouter un algorithme de description des points d'intérêts. Finalement, pour des raisons de comparaisons et pour réduire l'impact de la phase du recalage, c'est préférable d'utiliser le même algorithme que la majorité des méthodes.

Le détecteur de points d'intérêts SIFT est basé sur les étapes suivantes :

- **Création des pyramides Gaussian et DoG (Difference of Gaussian) :** Soit n_{pyr} la taille de la pyramide et Im_1 l'image sur laquelle on veut trouver les points d'intérêts. Pour créer le DoG, on définit les fonctions *reduce* et *expand* tel que la première réduit la taille d'une image par 2 et applique un filtre Gaussian et la deuxième augmente la taille d'une image par 2 et applique un filtre Gaussian. Le filtre Gaussien avec une déviation standard σ est défini par :

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1.10}$$

  - Une pyramide Gaussian P_G de taille n_{pyr} est créée en appliquant un filtre Gaussian $G(x, y, \sigma)$ avec la fonction *reduce* entre chaque niveau supérieur, ainsi les niveaux de la pyramide Pyr_G sont définis par l'équation 1.11.

$$Pyr_{G,i+1} = reduce(Pyr_{G,i}) \tag{1.11}$$

    Avec $i = 0...n_{pyr} - 1$ et $Pyr_{G0} = Im_1$.

  - Une pyramide DoG P_{DoG} de taille $n_{pyr} - 1$ est définie par la soustraction du niveau actuel d'une pyramide Gaussien avec son niveau inférieur après l'application de la fonction *expand* pour mettre le niveau inférieur à la même taille que le niveau actuel, l'équation 1.12 présente cette formulation. Cette pyramide est calculée d'une manière inverse, ce qui veut dire que l'image dans la base de la pyramide est la plus petite et l'image au niveau le plus haut a la plus grande taille.

$$Pyr_{DoG,i} = Pyr_{G,i} - expand(Pyr_{G,i-1}) \tag{1.12}$$

    Avec $i = n_{pyr} - 1...1$.

La figure 1.5 montre un exemple de pyramide Gaussian et de sa pyramide DoG.

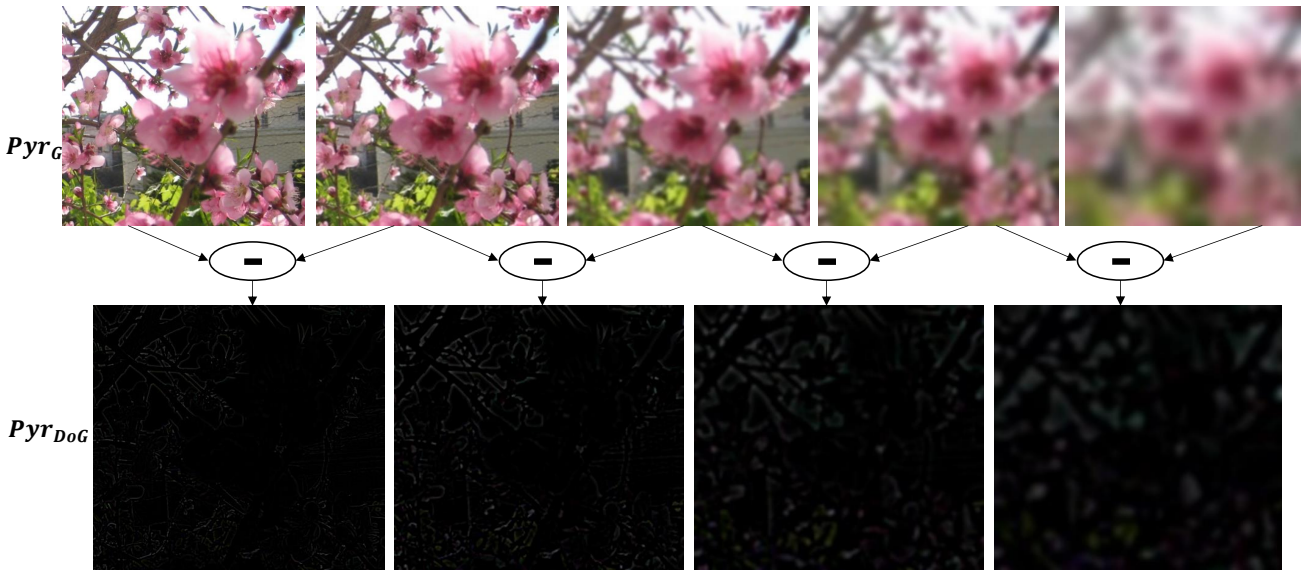


FIGURE 1.5 – Pyramide de taille $n_{pyr} = 5$ avec Pyr_G la pyramide Gaussian et Pyr_{DoG} la pyramide Laplacian/Difference of Gaussian

- **Sélection des points d'intérêts :** La sélection des points se fait en deux étapes
  - Premièrement, la fonction *expand* est appliquée sur toutes les images de la pyramide Pyr_{DoG} jusqu'à ce que tous les niveaux de la pyramide soit à la même taille que l'image originale. Ainsi, pour un niveau i, la fonction *expand* est utilisée i fois.
    L'objectif de cette opération est d'extraire les extrema en multi échelle pour créer un ensemble $E_{keypoints}$ où les points sont susceptibles à être des points d'intérêts. Par conséquence, chaque extrema est choisi par rapport aux 8 pixels de son voisinage de l'image dont il appartient, les 9 pixels à la même position de l'image inférieur à lui dans la pyramide et de l'image supérieur à lui pour un total de 26 points.
  - Deuxièmement, les points à faible contraste et les points de contours sont éliminés. Pour les points à faible contraste, ils sont éliminés si la magnitude de l'intensité de chaque point dans $E_{keypoints}$ est inférieur à un seuil, dans ce cas la formule de Taylor est utilisée pour les sous-pixels. Tandis que pour l'élimination des points de contours, une matrice Hessienne est calculée (en utilisant le filtre gradient dans la direction des abscisses et des ordonnées). Grâce à cette matrice, on enlève les points qui ont les gradients dans les deux directions qui sont faibles (régions planes), les points qui ont un gradient dans une direction qui est faible et élevé dans l'autre direction (contours) et on garde les points avec les gradients dans les deux directions qui sont élevés (coins).
- **Création des descripteurs :** Les descripteurs sont des représentations vectorielles des points sélectionnés dans $E_{keypoints}$. Chaque vecteur possède une taille de 128. Pour un point d'intérêts de $E_{keypoints}$, son descripteur est calculé de la façon suivante :
  - Tout d'abord, la magnitude et l'orientation sont calculées par rapport aux pixels voisins du point d'intérêt (la zone de voisinage est prix en fonction de la taille du Kernel du gradient choisi). Ensuite, un histogramme est créé et séparé en 36 bins (10 degrés par bin), par conséquence l'orientation et la magnitude du point d'intérêt est celle du pic de l'histogramme, si plusieurs pics existent alors de nouveaux points d'intérêts sont créés et ajoutés à l'ensemble. A noter qu'un pic est défini comme étant 80% à 100% de la valeur maximale.
  - Ensuite, une fenêtre de $(16, 16)$ est sélectionnée avec le point d'intérêt comme centre. Par conséquence, une interpolation doit être effectué pour les points autour. Par la suite, cette fenêtre est séparée en 16 sous-fenêtres de $(4, 4)$.
    Dans chaque sous-fenêtre, l'orientation et la magnitude sont calculées puis un histogramme séparé en 8 bins est calculé. De plus, un filtre de Gauss est appliqué pour que les pixels proches du point d'intérêt aient un plus grand impact sur la description.
    Cette opération est effectuée pour toutes les régions pour générer un vecteur de $4 \times 4 \times 8 = 126$. Ce vecteur est ensuite normalisé.

Ces trois étapes sont résumées par la figure 1.6.

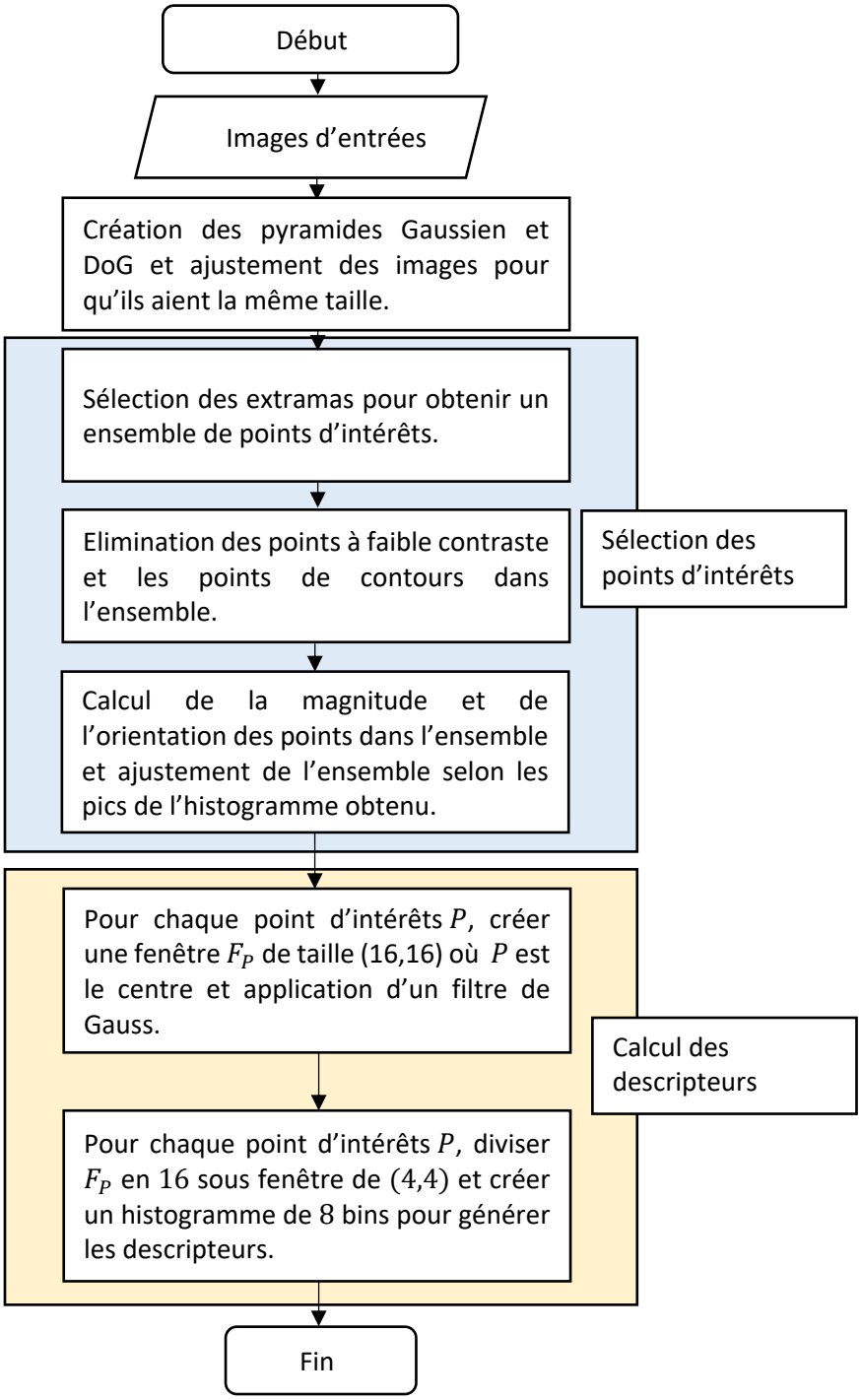


FIGURE 1.6 – Algorithme SIFT

1.3.4.2 Mise en correspondance et RANSAC

La mise en correspondance des points d'intérêts est une étape importante. En effet, une bonne mise en correspondance permet de réduire les distortions à la phase de la projection et les effets parallaxes durant l'assemblage des images. Ainsi, avec les points d'intérêts de deux images connus, cette étape permet de trouver les paires de points correspondants pour pouvoir calculer la transformation projective.

L'opération de mise en correspondance utilise les descripteurs qui sont des représentations vectorielles qui sont uniques aux points d'intérêts. Ainsi, l'objectif de cette dernière est de trouver les descripteurs d'une image qui correspondent aux descripteurs d'une autre image en évitant les points atypiques. Par conséquence, deux types de paires peuvent être distingués, les inliers qui sont des paires de points d'intérêts similaires et les outliers, des paires dissimilaires.

Il existe plusieurs moyens d'effectuer la mise en correspondance. D'une manière générale, la procédure de mise en correspondance est implémentée avec les étapes suivantes :

- Tout d'abord, une méthode de calcul de la distance est utilisée entre les descripteurs de la première image et les descripteurs de la seconde image.
- Ensuite, les paires sont formées avec les couples de descripteurs qui ont les distances les plus petites ou les plus grandes selon l'interprétation de similarité de la méthode de calcul de la distance.
- Finalement, les données aberrantes sont enlevées. Les paires avec une norme supérieure ou inférieure à un certain seuil choisi par l'utilisateur ne sont pas prises en considération.

La méthode de calcul de la distance peut être : une norme entre les deux vecteurs, la méthode de "Fast Library for Approximate Nearest Neighbors" (FLANN), la distance de Hamming, l'algorithme de comparaison "Brute Force" etc... D'autres méthodes, spécialement utilisées dans la création d'une mosaïque, existent comme par exemple, Laraqui et al. [36] qui ont utilisé un diagramme de Voronoi sur les points d'intérêts trouvés avec SIFT et ont comparé les zones avec la mesure de similarité "Zero Normalized Cross Corelation" (ZNCC) ou Yao et al. [37] qui ont utilisé une "Locality Sensitive Hashing" (LSH) pour indexer les descripteurs et créer des tables de hachages pour mesurer une similarité approximative afin d'extraire les paires similaires.

La mise en correspondance permet de trouver les quatre meilleurs paires pour calculer l'homographie, or l'utilisation de quatre paires n'est pas suffisant pour que la transformation projective soit optimale. En effet, vu que les paires trouvées après la mise en correspondance peuvent toujours contenir des données aberrantes, un algorithme qui estime les données pertinentes est utilisé à cette étape.

L'algorithme le plus utilisé dans le domaine des mosaïques est le "RANdom SAmple Consensus" (RANSAC) [38]. Cet algorithme prend d'une manière aléatoire 4 couples de points appariés pour calculer la matrice d'homographie initiale. Puis, il utilise cette homographie pour tester les autres couples appariés en comptant le nombre des "inliers" (c'est à dire un point multiplié par l'homographie donne une approximation du point correspondant). L'algorithme est effectué T fois. A chaque itération l'algorithme réalise la mise à jour de T grâce à la valeur de e qui est le ratio d'outliers qui est calculé en divisant le nombre de paires outliers à chaque itération de RANSAC par le nombre total de paires dans l'ensemble des couples appariés par la méthode de mise en correspondance après la description des points d'intérêts (Dans ce cas, c'est SIFT).

Des versions améliorés de RANSAC existent comme le "Maximum Likelihood Estimate SAmple Consensus" (MLESAC) [39] qu'on retrouve de temps en temps mais RANSAC reste toujours le plus populaire pour les méthodes de mosaïques d'images. La figure 1.7 montre les étapes de RANSAC.

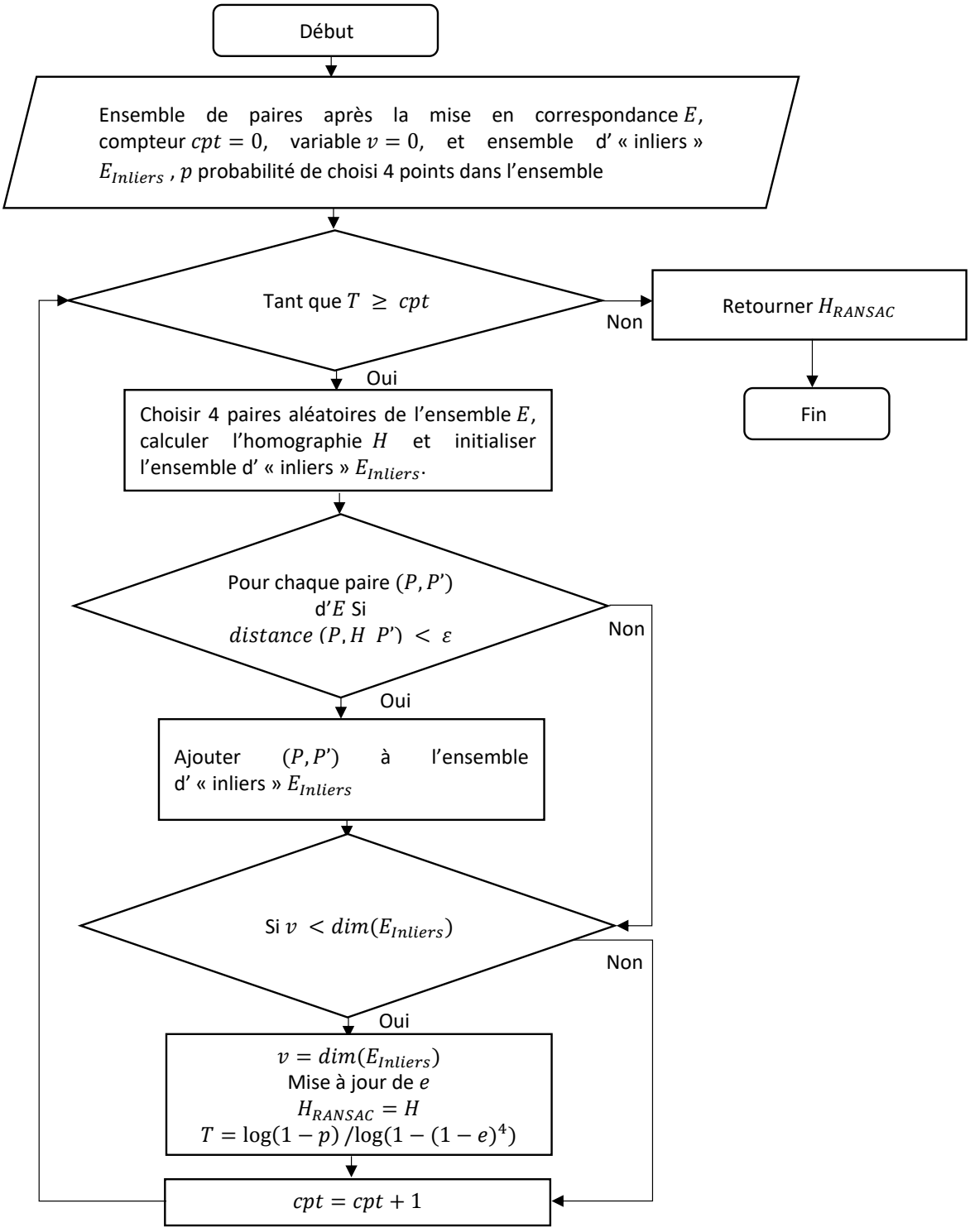


FIGURE 1.7 – Algorithme RANSAC

1.4 "Blending" d'images

Le "blending" d'images est une étape importante pour augmenter la qualité de l'image mosaïque. En effet, il permet d'enlever les discontinuités de radiométrie pour la génération d'un effet "seamless". En outre, lors de la phase d'assemblage, l'étape du "blending" calcule la nouvelle valeur des couleurs des pixels. Ainsi, l'objectif de cette dernière est de corriger les variations d'illumination qui surgissent à cause d'une différence d'exposure entre les images.

1.4.1 Méthodes de "blending" d'images

En général, il existe diverses méthodes pour effectuer le "blending" d'images [40]. En effet, trois catégories existent :

- **Méthodes basées sur le lissage des zones de transition** : Ces méthodes réduisent la différence radiométrique (couleur, intensité) dans la zone de transition entre la première et la deuxième image. Ce sont des méthodes qui parcourent la zone de chevauchement pixel par pixel et calculent une moyenne des valeurs des pixels, souvent avec un poids, pour l'attribuer à l'image finale.
- **Méthodes basées sur la recherche de la coupe optimale** : Ces approches parcourent l'image pour effectuer une coupure en trouvant le meilleur endroit pour l'avant-plan choisi sur l'arrière-plan. Elles sont souvent basées sur un algorithme "graphcut" ou un algorithme "watershed".
- **Méthodes hybrides** : Ces méthodes combinent entre les méthodes basées sur le lissage et les méthodes basées sur la recherche de la coupe optimale selon le besoin souhaité.

Soit I_1 et I_2 deux zones de chevauchement avec I_1 qui représente l'arrière-plan et I_2 l'avant-plan. On veut effectuer le blending de l'avant-plan sur l'arrière-plan pour obtenir l'image résultante I_f. Parmi les méthodes les plus utilisées dans le domaine des mosaïques d'images, on retrouve l' "alpha blending", le "feathering", le "multi band blending" et le "gradient domain blending". Dans certains cas, une combinaison de ces algorithmes est utilisée pour la génération d'un résultat optimal.

1.4.1.1 "Alpha blending"

L'"alpha blending" [41] est la méthode la plus intuitive et utilise un ratio de "blending" r_b dans la zone de transition (Equation 1.13), ce ratio est choisi par l'utilisateur ainsi si le ratio est élévé I_1 aura plus d'impact sur le résultat tandis que s'il est bas c'est I_2 qui aura plus d'impact. Ce "blending" est susceptible à créer des erreurs notamment s'il y a des erreurs de projections ce qui rendra la zone de transition du résultat floue.

$$I_f(x,y) = r_b \times I_1(x,y) + (1 - r_b) \times I_2(x,y) \tag{1.13}$$

Avec x et y les coordonnées d'un pixel.

1.4.1.2 "Feathering"

Le "feathering" [42] permet d'améliorer l'"apha-blending" en appliquant une transformée de distance $T(d)$ qui permet de calculer le ratio de "blending" d'une façon dynamique. En effet, un pixel central est choisi et tous les autres pixels de la zone de transition sont pondérés par rapport à ce pixel. La distance euclidienne est utilisée pour créer la carte de distance.

1.4.1.3 "Multi-band blending"

Le "Multi-band blending" ou le "pyramid blending" [43], utilise différentes pyramides pour assembler les images. C'est l'algorithme le plus utilisé pour effectuer le "blending" dans les mosaïques d'images. Ces étapes sont les suivantes :

- Premièrement, la zone où on veut effectuer le "blending" est choisie en créant une image masque I_m contenant des pixels soit avec une valeur de 0 ou de 255 où les pixels avec une valeur de 255 représentent la zone choisie.
- Deuxièmement, des pyramides Gaussians (Equation 1.11) sont créées pour les trois images (I_1, I_2 et I_m) et des pyramides Laplacians (Equation 1.12) pour les images d'origines (I_1 et I_2). En effet, en créant une pyramide Gaussian pour l'image masque, les bordures de valeur 255 vont être pondérées.
- Troisièmement, une pyramide P_s permettant d'assembler la pyramide masque et les pyramides Laplacians est calculé par l'équation 1.14.

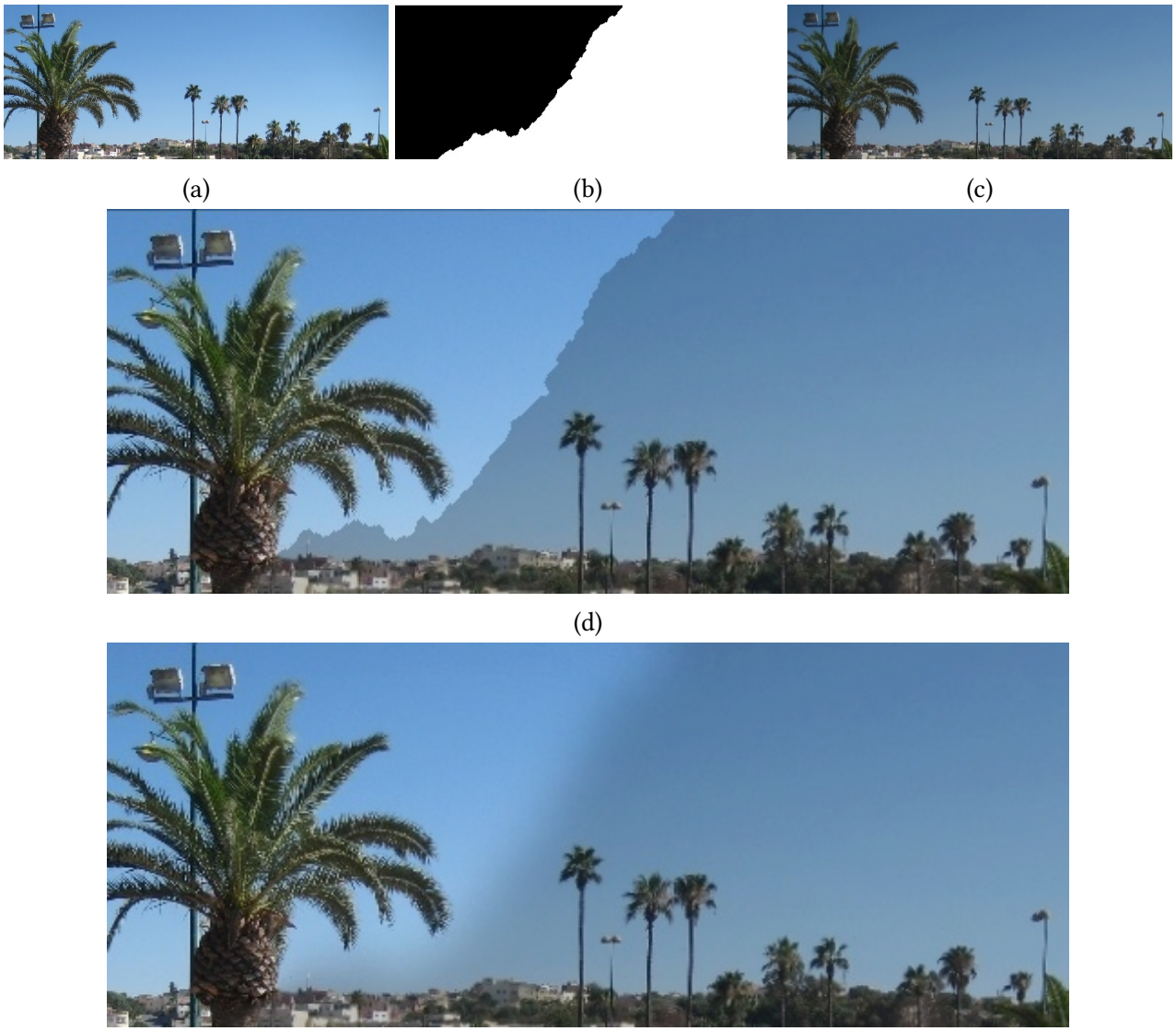
$$P_s^{i_{pyr}}(x,y) = P_{Gm}^{n_{blend}-1-i_{pyr}}(x,y) \times P_{L1}^{i_{pyr}}(x,y) + (1 - P_{Gm}^{n_{blend}-1-i_{pyr}}(x,y)) \times P_{L2}^{i_{pyr}}(x,y) \qquad (1.14)$$

  Où :
  - x et y sont les coordonnées d'un pixel.
  - i_{pyr} variable représentant le niveau de la pyramide avec n_{blend} le niveau maximum.
  - P_{Gm} la pyramide Gaussian de l'image masque I_m.
  - P_{L1} et P_{L2} les pyramides Laplacian des images I_1 et I_2 respectivement.
- Finalement, l'image résultante est reconstruite à partir de la pyramide P_s avec une somme de tous les niveaux après réajustement de leurs tailles avec la fonction $expand$.

La figure 1.8 présente un exemple de pyramide blending, dans l'image 1.8.d qui a été assemblée sans blending, la ligne où l'assemblage a été effectué est visible, contrairement à l'image 1.8.e.

1.4.1.4 "Gradient domain blending"

Le "Gradient domain blending" [44], aussi connu comme Poisson "blending", à l'origine il a été conçu pour insérer des objets dans des images sans obtenir de discontinuités dans le domaine de la composition d'images ("Image Compositing").



(a) (b) (c)

(d)

(e)

FIGURE 1.8 – Exemple de "multi-band blending" (d) Résultat sans blending (e) Résultat avec blending.

De la même manière que le "pyramid blending", le "Gradient domain blending" utilise une image masque pour choisir une zone où le "blending" est effectué. Il permet de calculer une correction aux pixels avec une valeur de 255 dans l'image masque par la résolution de l'équation de Poisson (Equation 1.15).

$$\nabla^2 I_f(x, y) = div(G_2) \tag{1.15}$$

Où :

- ∇^2 est le Laplacian.
- $I_f(x, y)$ représente la valeur du pixel de coordonnée (x, y) après l'application du blending, c'est les valeurs inconnus que l'on cherche.
- $div(G_2)$ représente la divergence du champ de vecteur Gradient de l'image foreground I_2.

L'équation 1.15 peut être résolue en respectant des conditions aux limites. Dans le Poisson "blending", la condition la plus utilisée est la condition aux limites de Newman [45] ("Newman's boundary conditions") même si dans certains cas on retrouve la condition aux limites de Dirichlet [45]. La condition aux limites de Newman permet de rendre toutes les valeurs Gradient avec une direction normal aux bordures égale à 0. Ainsi, à partir de l'équation 1.15, pour chaque pixel de l'image foreground I_2 avec une valeur de 255 dans l'image masque, on obtient l'équation 1.16, ce qui permettra de créer un système d'équations linéaires 1.17.

$$I_f(x+1,y) + I_f(x-1,y) + I_f(x,y+1) + I_f(x,y-1) - 4 \times I_f(x,y) \\ = G_{2x}(x,y) - G_{2x}(x-1,y) + G_{2y}(x,y) - G_{2y}(x,y-1) \quad (1.16)$$

Où G_{2x} et G_{2y} sont les gradients de l'image foreground I_2 par rapport aux axes des abscisses et ordonnées.

$$\underbrace{\begin{bmatrix} 4 & -1 & 0 & 0 & -1 & \dots & \dots \\ -1 & 4 & -1 & 0 & 0 & -1 & \dots \\ 0 & -1 & 4 & -1 & 0 & 0 & -1 \\ \dots & 0 & -1 & 4 & -1 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{\text{Matrice A}} \times \begin{bmatrix} I_f(i_P, j_P) \\ \vdots \end{bmatrix} = \begin{bmatrix} div(G_2(i_P, j_P)) \\ \vdots \end{bmatrix} \quad (1.17)$$

Où i_P, j_P sont des compteurs parcourant toutes les positions des pixels dans l'image foreground I_2 avec une valeur de 255 dans l'image masque.

L'équation 1.17 peut être écrite comme une équation de type $Ax = b$. La solution de cette équation peut être trouvée de plusieurs manières, en général on utilise la méthode Gauss-Seidel (Décomposition en en matrice triangulaire inférieur L et triangulaire supérieur U) ou la méthode des moindres carrés (Least square method) (Annexe A).

La solution obtenue à partir de l'équation 1.17 permet d'affecter les nouvelles valeurs des pixels de l'image foreground I_2 à l'image résultante I_f. Le reste des pixels de l'image résultante sont pris de l'image background I_1.

1.5 Prétraitement et Posttraitement

Le prétraitement et le posttraitement sont des traitements qui permettent de faciliter la création de la mosaïque dynamique et d'améliorer la qualité de l'image.

1.5.1 Prétraitement

Le prétraitement permet d'enlever le bruit, la différence de radiométrie pour faciliter le recalage d'images dans la détection de points d'intérêts, et dans certains cas pour améliorer la détection d'ob-

jet dynamiques. Il varie selon les différents objectifs qu'un domaine de recherche veut réaliser, par exemple le prétraitement pour une scène sous-marine [46] n'est pas le même pour une scène d'images ultrasons [47]. Dans le domaine des mosaïques d'images dynamiques, on retrouve le plus souvent la correction d'exposition comme prétraitement.

D'une manière générale, dans les systèmes qui utilisent les mosaïques d'images, deux types de prétraitements sont souvent recommandés : le premier étant le "bundle adjustment" [48] qui permet de classer les images d'une scène dans le même groupe dans le cas où les images d'entrées font partie de scènes différentes. Et le deuxième étant un algorithme pour le choix des images les plus pertinentes [49]. Ce dernier choisit les images qui créent le moins de problèmes parmi les images d'entrées ainsi que l'image la plus optimale qui va être utilisée comme image de référence pour le recalage d'images.

1.5.1.1 Correction d'exposition

La correction d'exposition ("Exposure correction") [50] est une méthode qui permet d'atténuer la différence d'exposition entre deux images Im_1 et Im_2 en calculant un gain et un biais tel que chaque paires de pixel inliers permet de résoudre l'équation 1.18.

$$Im_1(P_{inlier}) = \alpha \times Im_2(P^{'}_{inlier}) + \beta \tag{1.18}$$

Où :

- P_{inlier} et $P^{'}_{inlier}$ représente la position d'une paire de points inliers correspondants obtenu après le recalage d'images.
- α est le gain et β le biais.

Dans l'équation 1.18 les inconnus sont α et β. Vu qu'un ensemble d'inliers $E_{inliers}$ peut être obtenu après le recalage d'images (Section 1.3.4.1), un système d'équation peut être créé. Ce système d'équation est de type $Ax = b$ avec $x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ l'inconnu et il peut être résolu avec la méthode des moindres carrés (least square method) (Annexe A).

L'obtention du biais et du gain permet de modifier tous les pixels de l'image Im_2 pour que les pixels aient des intensités similaires à ceux de l'image Im_1. De plus, pour obtenir une meilleure compensation de gain, c'est préférable de calculer le gain et le biais pour chaque canal *RGB* en normalisant les valeurs d'intensités entre $[0, 1]$.

1.5.2 Posttraitement (Redressage d'images)

Le "blending" d'images permet déjà d'améliorer la qualité finale de l'image, ainsi il peut être considéré comme un posttraitement. Cependant, il effectue l'amélioration au niveau des couleurs pour avoir une continuité et un effet "seamless". Ainsi, le posttraitement utilisé dans le domaine des mosaïques consiste à améliorer la projection finale de l'image mosaïque. En effet, quand plusieurs images sont

utilisées durant la phase de recalage puis sont assemblées, un effet d'horizon courbé apparaît. La correction de cet effet se réalise avec un algorithme de redressage d'images [51] ("Image Straightening"). Ce dernier est un algorithme qui cherche les lignes horizontales les plus longues dans l'image puis effectue une rotation pour améliorer l'image, le degré de rotation à effectuer peut être calculé de plusieurs manières. Parmi les algorithmes de redressage, on retrouve deux méthodes qui sont les plus répandues, une qui est basée sur la transformée de Fourier rapide ("Fast Fourier Transform (FFT)") et la deuxième qui est basée sur la transformée de Hough. Dans cette sous section, on va présenter la seconde méthode.

La méthode basée sur la transformée de Hough permet de détecter les lignes droites dans une image. En appliquant cette transformée sur une image, on obtient une matrice d'accumulation H_{hough} où les abscisses et les ordonnées vont être les paramètres polaires (ρ, θ) avec $\theta \in [-90, 90]$. Les étapes du Straightening sont :

- Appliquer un détecteur de contours (filtre Prewitt, Canny, Sobel, etc...) sur l'image d'entrée Im_s pour obtenir une image binaire Im_c ;
- Utiliser la transformée de Hough sur Im_c pour obtenir la matrice d'accumulation H_{hough} ;
- Calculer la variance à chaque angle de la matrice H_{hough} pour obtenir le vecteur V_{hough} ;
- Séparer les variances d'angles en deux et les additionner tel que $V'_{hough}[i] = V_{hough}[i] + V_{hough}[i + 90]$ avec i = 0...89 ;
- Trouver l'angle θ_{angle} avec la variance V'_{hough} la plus élévée ;
- Appliquer une rotation à la mosaïque I_f avec un angle de $-\theta_{angle}$.

En général, le redressage est utilisé en posttraitement mais dans certains cas c'est préférable de l'utiliser en prétraitement sur les images d'entrées.

La figure 1.9 présente un exemple d'une mosaïque sans et avec redressage d'images.



(a)



(b)

FIGURE 1.9 – Mosaïque (a) Avant le redressage (b) Après le redressage avec un angle de 2.9° calculé avec Hough.

1.6 Conclusion

Dans ce chapitre, nous avons présenté et défini les mosaïques d'images dynamiques et les étapes pour leur création. Tout d'abord, nous avons commencé par le recalage d'images où nous avons exploré les méthodes les plus utilisées dans le domaine des mosaïques. L'algorithme SIFT a été expliqué en détails puisque c'est l'algorithme utilisé dans les chapitres suivants. Ensuite, les algorithmes de "blending" d'images ont été présentés avec une focalisation sur le "multi-band blending" et le "gradient domain blending". Finalement, le prétraitement et le posttraitement ont été discutés.

Chapitre 2

Traitement des objets dynamiques dans les mosaïques d'images

2.1 Introduction . . . 29
2.2 Artefacts . . . 29
2.3 Gestion des objets dynamiques . . . 30
2.3.1 Méthodes basées sur le calcul d'une "seamline" optimale . . . 32
2.3.1.1 Méthode de Mills et Dudek . . . 33
2.3.1.2 Méthode de Zeng et al. . . . 34
2.3.1.3 Méthode de Yao et al. . . . 36
2.3.1.4 Méthode de Li et al. . . . 37
2.3.2 Méthodes basées sur la reconstruction de régions . . . 39
2.3.2.1 Méthode d'Uyttendaele et al. . . . 39
2.3.2.2 Méthode de Bartoli et al. . . . 40
2.3.3 Comparaison des méthodes . . . 42
2.4 Evaluation des mosaïques . . . 42
2.4.1 Procédure d'évaluation . . . 43
2.4.2 Mesures quantitatives . . . 44
2.4.2.1 "Structure Similarity Index Measure" (SSIM) . . . 44
2.4.2.2 "Multi-Scale Structure Similarity Index Measure" (MS-SSIM) . . . 45
2.4.2.3 "Spectral Angle Mapper" (SAM) . . . 45
2.4.2.4 "Intensity Magnitude Ratio" (IMR) . . . 46
2.4.3 Evaluation des mosaïques d'images dynamiques . . . 46
2.5 Conclusion . . . 48

2.1 Introduction

Dans ce chapitre, les différentes catégories de méthodes utilisées pour le traitement des objets dynamiques dans le domaine des mosaïques seront présentées. En plus, plusieurs méthodes seront étudiées en détails. Mais avant d'entamer ces derniers, les objets dynamiques et les problèmes que nous souhaitons résoudre dans cette étape de création de mosaïques vont être définis. Finalement, le problème d'évaluation des mosaïques est traité et plusieurs moyens d'évaluations seront présentés.

2.2 Artefacts

Dans le domaine des mosaïques d'images, un objet dynamique peut être défini comme tout pixel pouvant créer une anomalie dans les zones de chevauchements, par conséquent un objet dynamique peut être un objet physique visible qui a changé de position entre les différentes zones communes des images ou une texture qui est différente d'une image à une autre soit parce qu'elle a subi un effet de mouvement invisible (par exemple modification de texture suite à un mouvement de vent) ou parce que durant la capture des images la source de lumière a changé.

Suite à une mauvaise gestion des objets dynamiques, plusieurs artefacts peuvent être observés dans la mosaïque finale dont certains sont impossible à corriger. En effet, il existe plusieurs erreurs qui peuvent entraîner à l'échec de la mosaïque, ces erreurs sont :

- **Les effets parallaxes :** Ces effets sont des distorsions qui apparaissent à cause d'une mauvaise projection des images. Ils peuvent être visibles comme ils peuvent être invisibles. En outre, dans le cas où ils sont visibles dans l'image, une discontinuité est créée et c'est souvent liée à une déformation de la zone, alors que dans l'autre cas la distorsion est moins visible car elle apparaît dans des zones homogènes et il faut zoomer pour la discerner. En plus, puisqu'une transformation projective contiendra toujours une quantité infime d'erreurs de projection, par conséquent quand plusieurs images d'entrées sont utilisées, l'effet parallaxe est plus probable d'apparaître à cause de l'accumulation d'erreurs de projection.
- **Le "ghosting" :** aussi connu sous le nom d'effet fantôme, c'est un effet que chaque objet dynamique dans la zone de chevauchement peut générer. Il est défini comme étant le reste d'un objet dynamique qu'un algorithme de gestion d'objet dynamique a mal traité. Entre autres, ça peut être une partie d'un objet dynamique, comme ça peut être les ombres de certains objets qui sont dur à traiter. Ainsi, plus la scène contient d'objets dynamiques plus cet effet est enclin d'apparaître.
- **Duplication d'objets :** effet qui survient dans la mosaïque résultante affichant une duplication d'informations par rapport aux images d'entrées. Cet effet est souvent non désiré sauf dans certains domaines d'applications où il est voulu pour faire une détection ou un suivi de mouvement. Le traitement de la duplication d'objets est délicat à traiter car en général, dans les mosaïques d'images dynamiques on traite les objets par pixel ou région et on ne cherche

pas à savoir si un objet qui était dans une certaine position est le même objet dans une autre position.

La figure 2.1 présente un exemple d'effet parallaxe et de ghosting.



(a)



(b)

FIGURE 2.1 – Artefact, (a) Effet parallaxe, (b) Ghosting.

Il existe d'autres erreurs qui sont plus globales avec un taux d'apparition élevé. En effet, c'est possible de prédire ces artefacts depuis l'acquisition des images d'entrées parmi eux on peut citer :

1) **Erreurs liées à la taille de la zone de chevauchement :** Quand la zone de chevauchement des images est petite. Dans ce cas, si la projection des images est réussie, il est fort probable que des effets parallaxes apparaissent et si un objet dynamique existe dans la zone, un ghosting va apparaître.
2) **Erreurs liées aux bordures de la zone de chevauchement :** Si les bordures de la zone de chevauchement passent à travers un objet dynamique alors un ghosting va forcément apparaître.
3) **Présence d'objets dynamiques physiques dans les zones non communes :** Cette présence va forcément provoquer une duplication d'objets sans aucun moyen de les contourner.
4) **Erreurs radiométriques :** une différence radiométrique entre les images accentue l'apparition d'effets parallaxes et ghosting.
5) **Erreurs liées aux occlusions :** Un objet dynamique qui est derrière un élément statique de l'image peut générer un ghosting plus facilement.

2.3 Gestion des objets dynamiques

L'étape de traitement des objets dynamiques se déroule généralement après le recalage d'images. Elle est effectuée sur la zone de chevauchement des images, ainsi c'est important d'extraire les zones de chevauchements avant de commencer le traitement. La zone de chevauchement de chaque image

est extraite en forme de rectangle en appliquant la transformation projective obtenue dans le recalage sur les quatres coins de deux images.

Il existe différentes méthodes pour détecter et gérer les objets dynamiques mais les étapes principales pour réaliser cette gestion sont similaires. En effet, deux étapes principales peuvent être observées, la première étant la détection d'objets dynamiques et la deuxième leur traitement pour créer le moins d'artefacts possible.

En cherchant dans le domaine des images à haute gamme dynamiques ("High dynamic range (HDR) images"), qui est un domaine proche et une des applications des mosaïques d'images dynamiques, on peut trouver que Tursun et al. [52] ont divisé les méthodes pour les images à haute gamme dynamiques en cinq catégories :

- "*Global exposure registration*" : règle les erreurs des mouvements en estimant les paramètres de transformation à appliquer pour chaque image en prenant en compte que les images d'entrées ne contiennent pas une différence d'exposure. Ils sont appliqués que pour régler les problèmes de mouvement de caméra lors de l'acquisition de la scène.
- "*Moving objet removal*" : Ces méthodes cherchent à séparer l'avant-plan de l'arrière-plan en utilisant toutes les images disponibles pixel par pixel.
- "*Moving object selection*" : Dans ces méthodes, les images sont sélectionnées pour détecter les objets dynamiques avec un algorithme puis les zones de chevauchements sont comparées. En général, une image ou plusieurs images sont utilisées comme référence pour chaque région dynamique.
- "*Moving object registration*" : Cette catégorie consiste à reconstruire les pixels considérés comme pouvant créant un artefact en utilisant une correspondance locale soit avec un flux optique ou des méthodes basées sur des zones"patch-based".
- "*Video deghosting*" : Cette catégorie utilise les mêmes méthodes que la catégorie de "Moving object registration" sauf que ces approches sont plus adaptées pour les vidéos.

Ainsi, en prenant en compte les catégories de Tursun et al. [52] et les différentes méthodes dans la littérature des années récentes, on a catégorisé les méthodes de gestion des objets dynamiques pour les mosaïques en deux catégories, on retrouve :

- Les méthodes basées sur le calcul d'une "seamline" optimale.
- Les méthodes basées sur la reconstruction de régions.

Chaque catégorie de méthodes possède des avantages et des inconvénients en fonction des images d'entrées et du résultat souhaité. Pour cette raison, certaines approches peuvent combiner entre les avantages de chaque catégorie pour créer des méthodes hybrides.

2.3.1 Méthodes basées sur le calcul d'une "seamline" optimale

Les méthodes basées sur le calcul d'une "seamline" optimale sont les plus populaires et en général se réalisent en deux étapes :

- La première étape consiste à créer une carte d'énergie. En effet, une carte d'énergie permet de surligner la position des pixels qui peuvent créer des artefacts. Elle est créée avec une équation qui permet de comparer la zone de chevauchement de deux ou plusieurs images. Ainsi, dans une carte d'énergie, les pixels avec une valeur proche de 0 vont représenter les zones similaires et les pixels avec une valeur élevée vont être des pixels qui peuvent créer des discontinuités.
- La seconde étape consiste à calculer une "seamline". Cette dernière est une coupure qui va séparer la zone de chevauchement des deux images. Avec la carte d'énergie de la première étape, le but c'est d'utiliser un algorithme programmé dynamiquement qui va chercher le chemin avec le coût le plus bas, c'est à dire que la "seamline" doit passer par un maximum de valeurs faibles en évitant les valeurs élevées. Ensuite, lors de la phase de l'assemblage, les pixels à gauche de la ligne de coupure vont être les pixels de la première image et ceux à droite vont être de la deuxième image. Cette procédure de programmation de "seamline" dynamique a été premièrement proposé par Duplaquet [53].

Ainsi, les méthodes de création de mosaïques d'images dynamiques qui utilisent la recherche de la "seamline" se diffèrent dans leur carte d'énergie et leur méthode de recherche de la coupure optimale. De plus, les méthodes dans cette catégorie ont tendance à traiter deux images à la fois. En outre, dans le cas de plusieurs images, une mosaïque temporaire est créée à partir des deux premières images, qui par la suite est considérée comme première image pour le prochain traitement jusqu'à ce que toutes les images soient traitées.

Il existe plusieurs méthodes dans cette catégorie. Parmi les méthodes on peut trouver des approches qui calculent des différences (intensité, gradient, texture etc...) pour créer une carte d'énergie afin de détecter les objets dynamiques. Par exemple, Mills et dudek [54] ont combiné entre une différence d'intensité et une différence de gradient, et Zeng et al. [55] ont utilisé la même combinaison mais ils ont ajouté un poids sur des pixels seuillés. Il y a aussi Yao et al. [37] qui ont combiné entre une différence de couleur et une différence de structure géométrique. De plus, la segmentation des objets dynamiques est un autre choix pour effectuer la détection des objets dynamiques. En effet, Li. et al [56] ont additionné plusieurs différences et ils les ont combiné avec une segmentation "meanshift" [57] pour trouver la position des images dynamiques et l'image dont ils appartiennent. D'une autre façon, Saito et al. [58] et Dong et al. [59] ont utilisé une segmentation sémantique pour trouver les zones fixes à éviter pour la création de mosaïques satellitaires/aériennes.

Par ailleurs, chaque méthode utilise un algorithme pour calculer une "seamline", ça peut être un algorithme de Dijkstra [54, 58], un "graphcut" [56], un algorithme "twin-snake" [60], un calcul d'une manière dynamique [37, 55] ou carrément un réseau pour chercher le meilleur chemin [61].

Dans cette sous-section on va présenter : la méthode de Mills et dudek [54], la méthode de Zeng et al. [55] nommée "Combined SIFT with dynamic programming" (CSDP), la méthode de Yao et al. [37] et la méthode de Li et al. [56] basée sur le "graphcut". Pour chaque méthode, on va définir la méthode de recalage d'images utilisée, la formule qui permet de calculer la carte d'énergie, l'algorithme utilisée pour calculer la "seamline" et l'approche de "blending" d'images utilisée.

2.3.1.1 Méthode de Mills et Dudek

La méthode de Mills et Dudek [54] est l'une des premières méthodes qui utilisent une "seamline" calculée pour la création d'une mosaïque. Cette méthode est utilisée avec deux images d'entrées à la fois. Tout d'abord, Mills et Dudek ont effectué le recalage d'images avec l'algorithme SIFT [24] (Section 1.3.4.1) et la méthode RANSAC [38] (Section 1.3.4.2).

Avant de détecter les objets dynamiques, Mills et Dudek ont utilisé une compensation de gain (Section 1.5.1.1) pour supprimer l'exposition entre les images. Ensuite, ils ont combiné entre une différence d'intensité (formulée avec l'équation 2.2) et une différence de gradient (Equation 2.3) pour créer la carte d'énergie. En effet, vu que la différence d'exposition entre les deux images n'est pas tout le temps linéaire, c'est probable que la correction ne fonctionne pas, c'est pourquoi ils ont additionné la différence d'intensité et la différence de gradient en utilisant le gain et le biais calculés dans l'étape de correction d'exposition. La formule (2.1) qui permet de calculer la carte d'énergie est la suivante :

$$\Delta_{xy} = w_1 \times \delta_{xy}^{Im} + w_2 \times \delta_{xy}^{\nabla} \tag{2.1}$$

Où :

- x et y sont les coordonnées des pixels.
- δ_{xy}^{Im} est la différence d'intensité de Im_1 et Im_2 qui représentent la zone de transition de chaque image. Cette différence d'intensité est calculée par l'équation 2.2 :

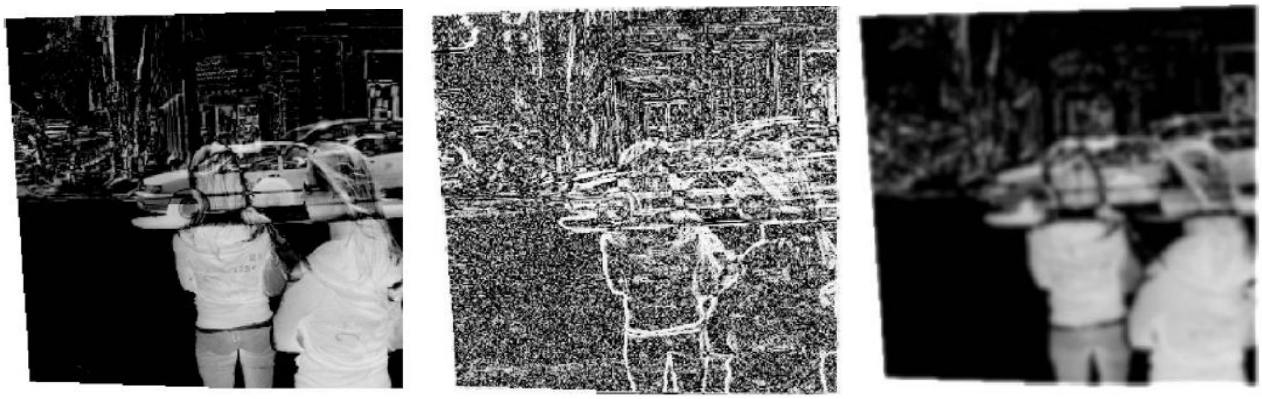
$$\delta_{xy}^{I} = \frac{abs(Im_{1xy} - Im_{2xy})}{max(Im_{1xy}, Im_{2xy})} \tag{2.2}$$

- δ_{xy}^{∇} est la différence des gradients de Im_1 et Im_2 avec Im_{G1} et Im_{G2} les gradients de Im_1 et Im_2. Elle est calculé par l'équation suivante :

$$\delta_{xy}^{\nabla} = \frac{abs(Im_{G1xy} - Im_{G2xy})}{max(Im_{G1xy}, Im_{G2xy})} \tag{2.3}$$

- w_1 et w_2 les poids de la carte d'énergie avec $w_2 = (abs(ln(\alpha)) + abs(\beta))^2$ et $w_1 = 1 - w_2$ où α est le gain et β le biais.

La figure 2.2 présente un exemple de Mills et Dudek [54] qui montre une différence d'intensité (Figure 2.2.a), une différence de gradient (Figure 2.2.b) et la carte d'énergie finale (Figure 2.2.c).



(a) (b) (c)

FIGURE 2.2 – Exemple de carte d'énergie de Mills et Dudek [54], (a) Différence d'intensité (Equation 2.2), (b) Différence de gradient (Equation 2.3), (c) Carte d'énergie (Equation 2.1)

Après avoir obtenu la carte d'énergie, les auteurs ont utilisé un algorithme de Dijkstra pour calculer la "seamline" dont la destination est le point avec la valeur la plus petite dans la dernière ligne de la carte d'énergie. Finalement, ils ont appliqué un "multi-band blending" (Section 1.4.1.3) pour obtenir une image mosaïque finale.

2.3.1.2 Méthode de Zeng et al.

Zeng et al. [55] se sont basés sur la méthode de Mills et Dudek [54] pour créer la méthode "Combined SIFT with Dynamic Programming" (CSDP). Ils ont suivi les mêmes étapes que Mills et Dudek. En effet, ils ont utilisé SIFT et RANSAC pour le recalage d'images, et ont utilisé le même algorithme de "blending", le pyramide blending. Cependant, ils ont amélioré la carte d'énergie calculée par l'équation 2.1 en ajoutant un seuil pour appliquer un poids aux valeurs élevées et ils ont calculé leur "seamline" d'une manière dynamique. De la même manière que la méthode précédente, CSDP n'utilise que deux images d'entrées à la fois.

Les différentes étapes pour le calcul de leur carte d'énergie sont les suivantes :

1) Au début, avant de calculer la carte d'énergie, ils ont appliqué un filtre Gaussian sur les images d'entrées de chevauchement Im_1 et Im_2.
2) Une différence d'intensité δ_{xy}^{Im} a été calculée par l'équation 2.2 de la même manière que la méthode de Mills et Dudek [54].
3) Une image masque I^{msk} a été calculée en utilisant δ^{Im} et un seuil $Thrd = 0.7$ tel que $I_{xy}^{msk} = 1$ si $\delta_{xy}^{Im} > Thrd$ sinon $I_{xy}^{msk} = 0$. Ensuite, une dilatation est appliquée sur I^{msk} pour obtenir $I^{msk'}$.
4) La nouvelle différence d'intensité $\delta^{Im'}$ est obtenue en remplaçant tous les pixels de δ^{Im} ayant une valeur $I_{xy}^{msk'} = 1$ avec un poids $weight$. Cette modification permet d'attribuer une valeur élevée aux pixels qui peuvent créer des artefacts.
5) La même procédure (Etapes 2, 3 et 4) est appliquée pour la nouvelle différence de Gradient $\delta^{\nabla'}$.

6) La carte d'énergie CSDP est calculée par l'équation 2.4. La figure 2.3 présente un exemple de Zeng et al. [54] de carte d'énergie où les éléments seuillés sont visibles.

$$\Delta_{xy} = w_1 \times \delta_{xy}^{Im'} + w_2 \times \delta_{xy}^{\nabla'} \tag{2.4}$$



FIGURE 2.3 – Exemple de carte d'énergie Zeng et al. [55], les éléments seuillés sont en vert.

La "seamline" de la méthode CSDP a été calculée d'une manière dynamique. En effet, le premier point de cette ligne de coupure est le point de la première ligne ayant la valeur la plus basse dans Δ, ensuite le point suivant est le point ayant la valeur la plus basse parmi les trois points de la ligne suivante du point actuel. Cette opération est effectuée jusqu'à ce que la dernière ligne soit atteinte. La figure 2.4 présente un exemple des pixels qui peuvent être choisis pour calculer une "seamline".

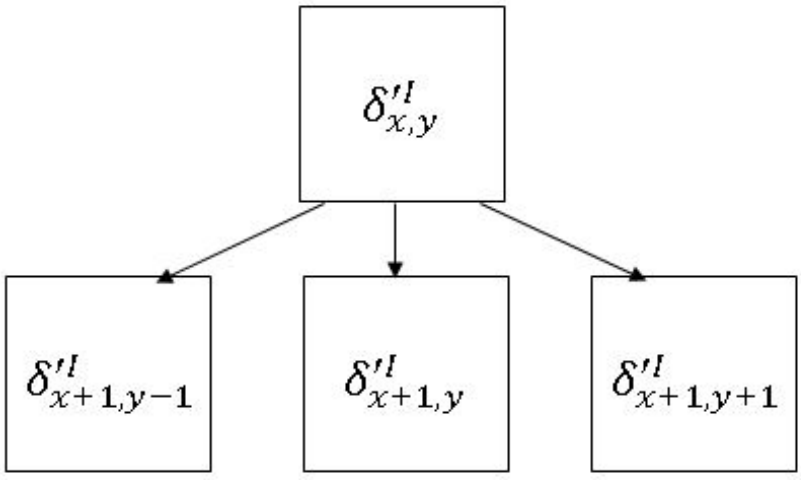


FIGURE 2.4 – Procédure de choix du prochain pixel pour la méthode CSDP.

Remarque : Zeng et al. [55] n'ont pas précisé l'ordre du choix des pixels dans le cas d'une zone homogène où les valeurs de la carte d'énergie ont la même valeur. Suite aux différents exemples dans

l'article, on a présumé que l'ordre est droite, milieu puis gauche. Ainsi, c'est cet ordre-là qui a été utilisé pour la programmation de cette méthode pour les prochains chapitres.

2.3.1.3 Méthode de Yao et al.

Yao et al. [37] ont commencé par un recalage modifié. En effet, ils ont extrait les points d'intérêts avec l'algorithme SURF [25], ensuite ils ont effectué une "Locality Sensitive Hashing" (LSH) pour raffiner les descripteurs avec un Hashing basé sur la distance de Hamming, finalement ils ont effectué la mise correspondance des points d'intérêts avec RANSAC [38].

La carte d'énérgie E de Yao et al. est basée sur deux paramètres, le premier étant la différence des intensités de couleurs et le deuxième sur la différence de la structure géométrique. Elle est calculée par l'équation 2.5.

$$E(x,y) = E_{color}(x,y)^2 + E_{geometry}(x,y) \tag{2.5}$$

Où :

- $E_{color}(x,y)$ est la différence d'intensités de couleurs avec $E_{color}(x,y) = Im_1(x,y) - Im_2(x,y)$
- $E_{geometry}(x,y)$ est la différence de structure géométrique avec $E_{geometry}(x,y) = Diff(Im_1(x,y) - Im_2(x,y))$ où la fonction $Diff$ est le produit de la différence entre le gradient des deux images par rapport aux axes des abscisses et des ordonnées.

Après le calcul de la carte d'énergie, Yao et al. ont choisi la "seamline" dans la zone de chevauchement qui a le coût total le moins élevé puis ils ont effectué leur "blending" en se basant sur la "seamline". En effet, pour un voisinage L, le "blending" est calculé par l'équation 2.6 pour chaque ligne des abscisses.

$$Im(x,y) = \begin{cases} Im_1(x,y) & x_{left} < x < x_{seam} - L/2 \\ \frac{d1}{L} \times Im_1(x,y) + \frac{d2}{L} \times Im_2(x,y) & |x - x_{seam}| < L/2 \\ Im_2(x,y) & x_{seam} - L/2 < x < x_{right} \end{cases} \tag{2.6}$$

Avec :

- Im le résultat final.
- Im_1 et Im_2 les images d'entrées.
- x_{left} la valeur des abscisses de la colonne la plus à gauche de la zone de chevauchement et x_{right} la colonne la plus à droite.
- x_{seam} la valeur des abscisses de la "seamline" dans la ligne courante.
- $d1$ la distance entre x_{seam} et x_{right}.
- $d2$ la distance entre x_{seam} et x_{left}.
- L une valeur pour déterminer le voisinage de x_{seam}.

La figure 2.5 explique la procédure en montrant les différents paramètres utilisés dans le "blending".

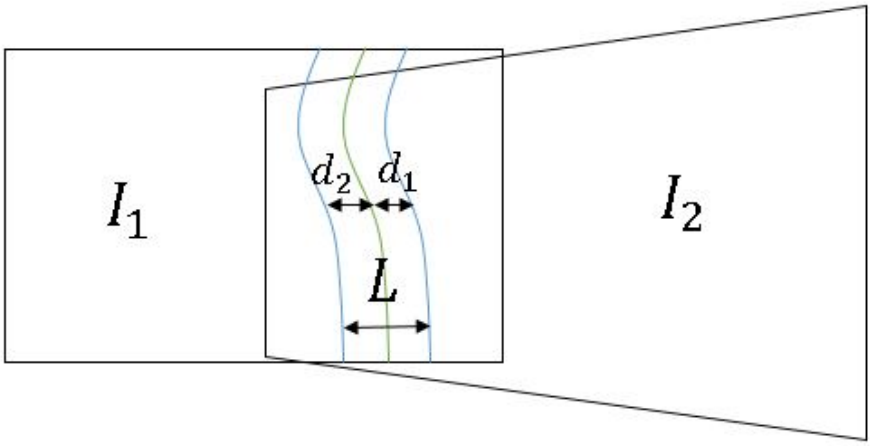


FIGURE 2.5 – Procédure de "blending" de la méthode de Yao et al. [37]

2.3.1.4 Méthode de Li et al.

Li et al. [56] ont créé une méthode basée sur le calcul de la "seamline" avec un "graphcut". Au début, ils ont créés leur carte d'énergie en combinant trois valeurs :

- Une différence d'intensité C_c calculée par l'équation 2.7.

$$C_c(x,y) = |Im_{g1}(x,y) - Im_{g2}(x,y)| \tag{2.7}$$

  Avec Im_{g1} et Im_{g2} les images aux niveaux de gris de la zone de chevauchement des images d'entrées Im_1 et Im_2.

- Une différence de gradient C_g calculée par l'équation 2.8.

$$C_g(x,y) = |G_{x1}(x,y) - G_{x2}(x,y)| + |G_{y1}(x,y) - G_{y2}(x,y)| \tag{2.8}$$

  Avec G_{x1} et G_{x2} les gradients de la zone de chevauchement par rapport à l'axe des abscisses des images d'entrées et G_{y1} et G_{y2} par rapport à l'axe des ordonnées.

- Une valeur C_t représentant la complexité de la texture. C_t est obtenue de la façon suivante :
  - Application d'un filtre Sobel sur chaque image dans les deux axes pour obtenir S_x et S_y.
  - Calcul des angles et magnitudes de chaque pixel en utilisant S_x et S_y.
  - Création de l'histogramme de gradients (HoG) avec un nombre de bins B avec les angles obtenus comme axe et les magnitudes comme valeur pour chaque image.
  - Calculer C_t par l'équation 2.9.

$$C_t(x,y) = \Gamma_1(x,y) + \Gamma_2(x,y) \tag{2.9}$$

    Avec : $\Gamma(x,y) = 1 - \frac{\sum_{b=1}^{B} min(H_b(x,y)),\overline{H}(x,y)}{\sum_{b=1}^{B} H_b(x,y)}$
    Où : H est l'histogramme d'objet orienté pour un bin b et $\overline{H}$ la moyenne de l'histogramme.

- La carte d'énergie finale est obtenue par l'équation 2.10.

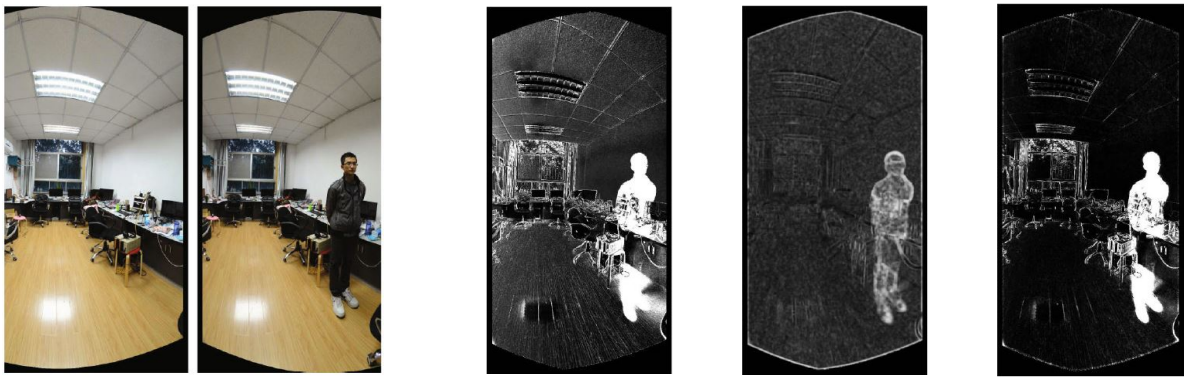
$$C(x,y) = (C_c(x,y) + C_g(x,y)) \times C_t(x,y) \tag{2.10}$$

Après l'obtention du coût d'énergie, ils ont détecté les objets dynamiques en utilisant une carte de distance et une segmentation "meanshift" [57]. En effet, la carte de distance D a été calculée par l'équation 2.11. La figure 2.6 montre un exemple de calcul de la carte de distance. Cette carte a permis de détecter les régions dynamiques (O). Ensuite, une segmentation "meanshift" a été utilisée sur les images d'entrées et chaque région dynamique a été comparée avec les régions obtenues par la segmentation pour calculer les probabilités P_1 et P_2 où P_1 est la probabilité qu'un pixel appartient à la première image et P_2 qu'il appartient à la seconde image.

$$D(x,y) = -\frac{1}{ln(D_h(x,y))} \times D_c(x,y) \tag{2.11}$$

Avec :

- D_c la somme de la différence au carré entre chaque canal des images d'entrées après leur conversion dans l'espace CieLab.
- D_h la distance des histogrammes de gradient entre H_1 et H_2 des images Im_1 Im_2 tel que $D_h(x,y) = \sqrt{1 - \frac{1}{\sqrt{\overline{H}_1(x,y) \times \overline{H}_2(x,y) \times B}} \times \sum_{b=1}^{B} \sqrt{H_1^b(x,y) \times H_2^b(x,y)}}$



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

FIGURE 2.6 – Exemple de carte de distance Li et al. [56], (a) Images d'entrées, (b)Différence au carré des canaux CIELab D_c, (b) Différence des histogrammes de gradient HoG D_h, (c) Carte de distance D(Equation 2.11)

La "seamline" de Li et al. a été calculée par la méthode "Graphcut" en utilisant une carte d'énergie modifiée (Equation 2.12) calculée avec les anciennes valeurs de la carte d'énergie et la probabilité d'apparition dans une image.

$$E(\mathcal{I}) = E_{data}(\mathcal{I}) + E_{smooth}(\mathcal{I}) \tag{2.12}$$

Avec :

- $\mathcal{I}$ une région composite des images d'entrées dans la zone de chevauchement parmi les régions détectés.

- $E_{data}(\mathcal{I}) = \sum_{(x,y)\in\mathcal{I}}(D_l^1(x,y) + D_l^2(x,y))$
  Avec $D_l(x,y) = P(O) \times T$ si (x,y) appartient à la zone de transition sinon il est égal 0 et $T = 100$ une pénalité.
- $E_{smooth}(\mathcal{I}) = \sum_{(P_1,P_2)\in\mathcal{N}((I))} \sigma(P_1,P_2) \times (C(P_1) + C(P_2))$
  où P_1 et P_2 sont deux pixels adjacents dans le voisinage $\mathcal{N}$ et $\sigma(P_1,P_2) = 0$ si P_1 et P_2 sont similaires sinon il est égal à 1 .

2.3.2 Méthodes basées sur la reconstruction de régions

Les méthodes basées sur la reconstruction de régions sont des méthodes qui traitent les objets dynamiques d'une manière plus globale que les méthodes basées sur la recherche d'une "seamline" optimale. En effet, premièrement, la détection des objets dynamiques est effectuée. Ce processus est réalisé par régions, où chaque région représente un ensemble de pixel pouvant créer des artefacts. Les pixels dynamiques sont premièrement détectés et ceux qui sont adjacents sont considérés comme une région. Deuxièmement, il existe deux manières de traiter les régions dynamiques détectées. D'une part, pour la première façon de traitement, les régions sont reconstruites en choisissant le meilleur résultat possible comme c'est le cas pour Uyttendaele et al. [62] qui ont généré des régions de différences et ont reconstruit la mosaïque finale avec un graphe qui produit le meilleur résultat ou Qu et al. [63] qui ont séparé les images en blocs et ont reconstruit la mosaïque après le traitement de chaque bloc. De l'autre part, la deuxième façon pour traiter les objets dynamiques détermine les régions arrière-plan par rapport aux régions avant-plan pour reconstruire la mosaïque selon le besoin soit pour une reconstruction de scène sans avant-plan ou pour un suivi de mouvement. Dans la littérature, Bartoli et al. [17] ont reconstruit leur mosaïque grâce à une distance de Mahalanobis normalisée qui leur a permis de déterminer l'arrière-plan et l'avant-plan après avoir créé une matrice de projection améliorée tandis que Zhi et al. [64] ont créé une méthode basée sur la profondeur pour extraire l'avant-plan de l'arrière-plan.

Dans cette sous-section, on va présenter la méthode Uyttendaele et al. [62] et la méthode Bartoli et al. [17].

L'un des avantages de cette catégorie c'est la possibilité de traiter les objets dynamiques avec plus de deux images en même temps.

2.3.2.1 Méthode d'Uyttendaele et al.

Uyttendaele et al. [62] ont créé des régions de différence RoD ("Regions of Difference") à partir de chaque zone de chevauchement entre les différentes images. En effet, pour chaque paire d'images, une zone de chevauchement est obtenue et chaque zone génère une RoD. Dans la zone de chevauchement, les pixels sont comparés un à un, si la différence de la valeur d'intensité au niveau de gris est inférieur à un seuil, ce pixel est ajouté à la RoD de cette zone.

Après la création de RoD, pour chaque zone de chevauchement, Uyttendaele et al. ont généré un

graphe où chaque nœud est une image et les nœuds sont reliés avec des liens qui sont les RoDs. Ainsi, deux images reliées avec un lien possèdent une zone de chevauchement, et donc ils possèdent une RoD où il faut prendre la décision de quelle image utiliser pour reconstruire la zone. Tandis que les images non reliées dans le graphe sont des images qui ne possèdent pas une zone commune. La figure 2.7 présente un exemple de création de graphe RoD à partir de 4 images.

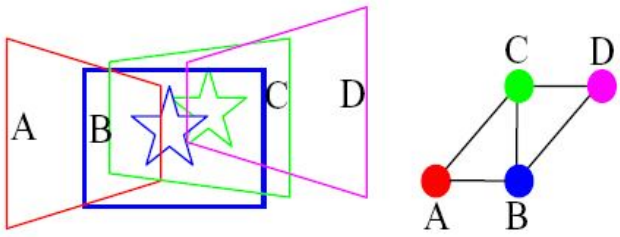


FIGURE 2.7 – Exemple de création de graphe RoD d'Uyttendaele et al. [62]

Pour créer la mosaïque finale, la méthode parcourt le graphe avec la première image comme nœud de départ et la dernière image comme nœud d'arrivée. Afin de trouver le meilleur chemin, un poids est attribué aux liens. Ce poids est basé sur le nombre de pixels dans un RoD et sur leur position par rapport au centre de l'image. En effet, dans un RoD, plus les pixels détectés sont proche du centre de l'image finale plus ils auront une valeur élevée. Finalement, l'image résultante est reconstruite en choisissant le chemin qui a le coût le plus bas.

Afin d'améliorer l'intégration des régions reconstruites dans la mosaïque finale, Uyttendaele et al. [62] ont utilisé un moyenneur avec un Kernel de $(1/4, 1/2, 1/4)$ en deux itérations puis ont appliqué un "blending" pour améliorer l'exposure en utilisant une interpolation bilinéaire ou bicubique. Cette opération a été réalisée en séparant la mosaïque en blocs (de préférence de taille $(32, 32)$) et en appliquant l'équation 2.13.

$$Im(x,y) = \sum_{v=-1}^{1} C_v(\frac{y}{N}) \sum_{u=-1}^{1} C_u(\frac{x}{N}) f_{m+u,n+v}(Im(x,y)) \tag{2.13}$$

Où :

- x et y sont les coordonnées à l'intérieur du bloc N.
- C_n le coefficient bilinéaire ou bicubique avec $n \in u, v$.
- $f_{m,n}$ fonction de transfert.

2.3.2.2 Méthode de Bartoli et al.

Bartoli et al. [17] ont créé une homographie améliorée H_{imp} pour effectuer le recalage d'images. Au début, Ils ont détecté les points d'intérêts avec Harris. Puis, en se basant sur les paramètres intrinsèques et extrinsèques de la caméra, ils ont supposé que la distance focale de deux images consécutives ne varient pas (9a ne veut pas dire qu'elle est constante) et ainsi ils ont réduit le nombre d'inconnus dans l'homographie de 8 à 4. Ensuite, ils ont utilisé, MSAC ("M-estimator SAmple and Consensus")

(une version de RANSAC) pour trouver les "inliers". La méthode de Bartoli et al. [17] est adaptée pour les vidéos ou plusieurs images consécutives. Les étapes sont les suivantes :

- Pour chaque image, l'homographie améliorée est calculée entre l'image actuelle et l'image suivante $H^{-1}_{i+1,i}$ et l'image actuelle et l'image précédente $H_{i,i-1}$.
- Chaque pixel dans une des 3 images est comparé avec une distance de Mahalanobis normalisée (Equation 2.14) avec les pixels à la même position dans les deux autres images (Valeurs entre 0 et 1) et une probabilité p est calculée (Equation 2.15) pour estimer si le pixel fait partie de l'avant-plan ou l'arrière-plan. Si la probabilité p est élevée, alors le pixel fait partie de l'avant-plan.

$$d^2(Im_i(x,y), Im'_{i-1}(x,y)) = (Im_i(x,y) - Im'_{i-1}(x,y))^T \times C^{-1} \times (Im_i(x,y) - Im'_{i-1}(x,y)) \quad (2.14)$$

  Où :
  - Im est l'image d'origine, Im' est l'image après projection.
  - x et y les coordonnées du pixel.
  - C la matrice de covariance de chaque canal de toutes les images.

$$p_{x,y} = max(d_N(Im_i(x,y), Im'_{i-1}(x,y)), d_N(Im_i(x,y), Im'_{i+1}(x,y))) \quad (2.15)$$

  Avec d_N la probabilité normalisée.
- Afin d'éviter les problèmes obtenus à cause des zones homogènes et des objets dynamiques qui changent en restant à la même position, un filtre Gaussian a été appliqué sur la carte de probabilité p. Puis, la probabilité la plus grande p_{max} est récupérée pour seuiller la carte de probabilité en mettant toutes les valeurs inférieures à $p_{max}/2$ comme faisant partie de l'arrière-plan. La figure 2.8 montre un exemple des différentes étapes de la carte de probabilité ainsi que l'avant-plan obtenu par chaque carte.
- La valeur finale associée au panorama est obtenue avec l'équation 2.16, elle est calculée avec seulement les images qui font partie de l'arrière-plan.

$$Im(x,y) = \sum_{l=1}^{k} \frac{1}{\sqrt{2\pi\sigma}} \times exp(\frac{(Im_l(x,y) - \mu)^2}{2\sigma^2}) \times Im_l(x,y) \quad (2.16)$$

  Avec :
  - x, y les coordonnées du pixel.
  - k le nombre total de pixels appartenant à l'arrière-plan à la position (x, y).
  - μ la moyenne des valeurs des pixels appartenant à l'arrière-plan.
  - σ l'écart type des valeurs des pixels appartenant à l'arrière-plan.

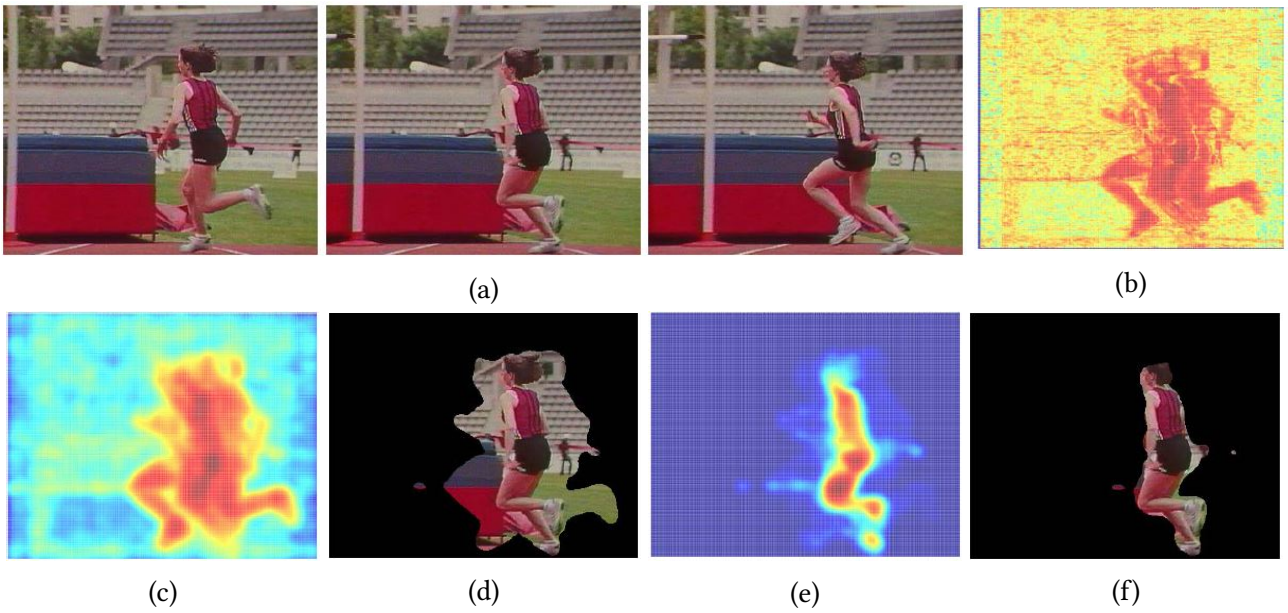


FIGURE 2.8 – Exemple de la méthode Bartoli et al. [17], (a) Images d'entrées, (b) Carte de probabilité p, (c) Carte de probabilité après filtre Gaussian, (d) Avant-plan de l'image d'entrée du milieu obtenue avec la carte de probabilité après filtre Gaussian (e) Carte de probabilité après seuillage (f) Avant-plan de l'image d'entrée du milieu après seuillage.

2.3.3 Comparaison des méthodes

Chaque méthode de création de mosaïque possède des points forts et des points faibles. La majorité des méthodes basées sur le calcul des "seamlines" sont plus adaptées pour les scènes qui contiennent beaucoup d'objets dynamiques, ils ont aussi une meilleur chance d'éliminer les effets parallaxes, par contre ils génèrent plus facilement une duplication d'objets et du "ghosting" si des occlusions existent dans les images d'entrées. De plus, les méthodes basées sur le calcul des "seamlines" sont généralement plus lentes que les autres méthodes vu qu'ils traitent les images par paires, ainsi un grand nombre d'images d'entrées ralentit la création de la mosaïque.

Concernant les méthodes basées sur la reconstruction de régions, ils éliminent plus facilement les duplications d'objets et sont plus robustes aux erreurs qui peuvent apparaitre à cause d'occlusions. Cette catégorie traite les images avec un nombre élevé contrairement aux méthodes basées sur le calcul d'une "seamline", ainsi les méthodes sont plus rapides. Cependant, afin de reconstruire la mosaïque d'une manière optimale, ces méthodes ont besoin qu'à un certain moment dans le temps que toutes les parties de l'image doivent posséder un arrière-plan. Ainsi, si les images d'entrées contiennent trop d'objets dynamiques et une zone de la mosaïque ne possède pas une image d'entrée avec cette zone qui est statique, alors un artefact va forcément apparaître.

2.4 Evaluation des mosaïques

Evaluer une image mosaïque est une tâche difficile et l'évaluation en elle-même est un domaine de recherche. En effet, une mosaïque résultante est une image synthétisée. Ainsi, chaque mosaïque

obtenue est unique par rapport à la méthode utilisée pour la créer. De ce fait, deux méthodes différentes utilisant les mêmes images d'entrées ne peuvent créer le même résultat. C'est ce qui rend l'évaluation d'une mosaïque une étude délicate vu qu'une mosaïque de référence de bonne qualité utilisée comme un point de repère n'existe pas.

L'un des atouts des mosaïques qui les rendent facilement comparable pour un être humain, c'est le résultat visuel. Supposons que nous souhaitons comparer N mosaïques créées avec M méthodes (Chaque mosaïque possède M résultats). Il suffit de donner un score $\in [1, M]$ non répétitives à chaque résultat obtenu par chaque méthode. Ensuite, répéter ceci pour toutes les N mosaïques. Et finalement, calculer une moyenne des scores pour chaque méthode pour déterminer la meilleure méthode ayant le score le plus élevé. Cependant pour une machine, réaliser cette opération est compliqué surtout que des méthodes différentes qui ont créé des mosaïques différentes peuvent tous donner des résultats qui sont corrects.

Scientifiquement, donner un score pour évaluer une mosaïque d'une manière visuelle n'est pas acceptable. Une mesure quantitative doit être calculée. Dans cette section, on va présenter la procédure d'évaluation des mosaïques, ainsi que différentes mesures quantitatives.

2.4.1 Procédure d'évaluation

Puisque chaque méthode crée un résultat unique, mesurer la qualité des mosaïques entre eux peut être difficile. Ainsi, pour pouvoir calculer une valeur quantitative permettant de représenter la qualité d'un résultat, la mosaïque d'une méthode est comparée avec les images d'entrées.

L'objectif de cette comparaison est de vérifier si l'image mosaïque finale a respecté les images d'entrées sans les distordre. De ce fait, les mosaïques avec une mesure élevée vont être les plus proches des images d'entrées. Cependant, avec la présence d'objets dynamiques, une manière alternative est effectuée pour calculer la mesure.

Soit Im_1 et Im_2 les images d'entrées après le recalage d'images et Im_R la mosaïque résultante d'une méthode.

La procédure pour évaluer une mosaïque est la suivante :

- La première étape consiste à extraire la zone de chevauchement I_1 et I_2 des images Im_1 et Im_2, ainsi que I_R la zone de chevauchement du résultat d'une méthode de création de la mosaïque Im_R. La comparaison de Im_R avec Im_1 et Im_2 est effectuée sur la zone de chevauchement de Im_1 et Im_2 puisque c'est là où les changements existent dans la mosaïque résultante.
- La seconde étape consiste à calculer une mesure de comparaison pour chaque pixel entre I_R et I_1 et entre I_R et I_2, puis à choisir la meilleure valeur. Ainsi, On définit une fonction temporaire f pour mesurer une valeur de comparaison, avec f_g la fonction de comparaison entre I_R et I_1 et f_d la fonction de comparaison entre I_R et I_2. Pour chaque pixel (x, y) dans la zone

de chevauchement la valeur finale d'évaluation de ce pixel est la valeur optimale entre f_g et f_d selon l'interprétation de la fonction de comparaison. Cette opération permet également de déterminer l'origine du pixel dans la mosaïque.

- La dernière étape consiste à calculer la moyenne des mesures obtenues de chaque pixel.

Dans le cas d'une mosaïque générée avec plus de deux images, la mesure de comparaison est calculée par rapport aux nombres d'images présentes dans les zones de chevauchements avant de choisir la valeur optimale. Par exemple pour 3 images c'est possible d'avoir au maximum 3 zones de chevauchements : une zone avec la première et la seconde image, une zone avec la seconde et troisième image et une zone avec toutes les images.

2.4.2 Mesures quantitatives

Il existe différentes mesures quantitatives qui peuvent être utilisées dans la comparaison des mosaïques. Dans cette sous-section, on va tout d'abord présenter la "Structure Similarity Index Measure" (SSIM) [65, 66], puis la "Multi-Scale Structure Similarity Index Measure" (MS-SSIM) [67] qui est une SSIM à plusieurs échelles et finalement le "Spectral Angle Mapper" (SAM) [68] et l'"Intensity Magnitude Ratio" (IMR) [69].

2.4.2.1 "Structure Similarity Index Measure" (SSIM)

La "Structure Similarity Index Measure" (SSIM) [65] est un index qui mesure la similarité des composantes géométriques entre deux images. Il est obtenu en calculant 3 composantes exprimées par l'équation 2.17. Ces trois composantes sont la luminance (Equation 2.18), le contraste (Equation 2.19) et la structure (Equation 2.20). L'une des particularités de la SSIM c'est le faite que la valeur d'un pixel est calculée sur un bloc (en général de taille $(32, 32)$), ce qui permet de prendre en compte le voisinage du pixel dans la comparaison.

$$SSIM(a,b) = l(a,b)^{\alpha} \times c(a,b)^{\beta} \times s(a,b)^{\gamma} \tag{2.17}$$

Avec :

$$l(a,b) = \frac{(2 \times \mu_a \times \mu_b + c_1)}{(\mu_a^2 + \mu_b^2 + c_1)} \tag{2.18}$$

$$c(a,b) = \frac{(2 \times \sigma_a \times \sigma_b + c_2)}{(\sigma_a^2 + \sigma_b^2 + c_2)} \tag{2.19}$$

$$s(a,b) = \frac{(\sigma_{ab} + c_3)}{(\sigma_a \times \sigma_b + c_3)} \tag{2.20}$$

Où :

- a et b sont deux pixels.
- μ_a et μ_b sont les moyennes des blocs centrés sur les pixels a et b.
- σ_a et σ_b sont les écarts types des blocs centrés sur les pixels a et b.
- σ_{ab} la covariance entre les blocs centrés sur les pixels a et b.
- $c_1 = (k_1 \times L)^2$, $c_2 = (k_2 \times L)^2$ et $c_3 = c_2/2$ sont des constantes. Par défaut $k_1 = 0.01$, $k_2 = 0.03$, $\alpha = \beta = \gamma = 1$ et L est la valeur maximale qu'un pixel peut atteindre.

SSIM retourne une valeur entre $[-1, 1]$. Plus la valeur est proche de 1 plus les deux éléments comparés sont similaires. La comparaison est considérée comme optimale si la valeur retournée est supérieur à 0.98.

2.4.2.2 "Multi-Scale Structure Similarity Index Measure" (MS-SSIM)

La "Multi-Scale Structure Similarity Index Measure" (MS-SSIM) [67] est calculée de la même manière que la SSIM sauf qu'elle est effectuée à plusieurs échelles. Afin de calculer la MS-SSIM, une fonction *reduce*, qui réduit la taille de l'image par 2 et utilise un filtre passe bas en même temps, est appliquée K fois sur le paramètre de contraste et de structure d'une manière itérative. (Equation 2.21)

$$MS_SSIM(a, b) = l(a, b)^{\alpha} \times \prod_{k=1}^{K} c_k(a, b)^{\beta} \times s_k(a, b)^{\gamma} \tag{2.21}$$

De la même manière que SSIM, MS-SSIM retourne des valeurs entre $[-1, 1]$ où une comparaison est considérée comme optimale si la valeur retournée est supérieur à 0.98.

2.4.2.3 "Spectral Angle Mapper" (SAM)

Le "Spectral Angle Mapper" (SAM) [68] est un index qui compare la qualité de deux vecteurs de couleurs. Il retourne un degré qui est calculé avec un *arccos* d'un produit vectorielle divisé par la norme des deux vecteurs (Equation 2.22). L'un des avantages de SAM c'est qu'il permet de calculer la similarité sans être influencer par la valeur de l'intensité.

$$SAM(\vec{a}, \vec{b}) = arccos(\frac{\vec{a}.\vec{b}}{(||\vec{a}|| \times ||\vec{b}||)}) \tag{2.22}$$

Où $\vec{a}$ et $\vec{b}$ sont deux vecteurs de couleurs.

Un angle élevé entre deux vecteurs de couleurs est interprété comme une dissimilarité entre les deux vecteurs. Un angle permettant de prouver une similarité optimale doit être inférieur à 2 degrés.

2.4.2.4 "Intensity Magnitude Ratio" (IMR)

L'"Intensity Magnitude Ratio" (IMR) [69] est une mesure de comparaison simple qui compare la qualité des intensités. Sa formule est présentée par l'équation 2.23.

$$IMR(a,b) = \frac{min(|x|,|y|)}{max(|x|,|y|)} \tag{2.23}$$

IMR retourne des valeurs entre 0 et 1. Plus les valeurs sont élevées, plus les intensités sont similaires. Une valeur est considérée comme optimale si elle est supérieure à 0.98.

2.4.3 Evaluation des mosaïques d'images dynamiques

D'autres méthodes existent pour le calcul de la qualité d'une image. Parmi eux : l'utilisation des méthodes NR-IQA ("No-Reference Image Quality Assessment"). Les NR-IQA mesurent la qualité d'une image en la comparant avec un modèle généré à partir de plusieurs images d'entraînements. En effet, l'algorithme choisit des images aléatoires à partir des différentes images entraînements, ensuite il utilise une méthode pour extraire les caractéristiques des images séléctionnées, puis il applique une méthode de machine à vecteurs de support pour résoudre la régression ("Support Vector Regression" (SVR)) pour finalement obtenir le modèle de qualité.

Pour calculer la mesure de qualité d'une image avec une méthode NR-IQA, les caractéristiques (points d'intérêts, descripteurs, histogrammes, etc...) de l'image sont extraites et elles sont comparées avec le modèle en utilisant le SVR pour obtenir le score de l'image. En outre, les méthodes utilisées sur l'image pour mesurer sa qualité doivent être les mêmes méthodes qui ont été utilisées pour générer le modèle de qualité. Plusieurs mesures existent, les mesures "Blind/Referenceless Image Spatial Quality Evaluator" (BRISQUE), "Naturalness Image Quality Evaluator" (NIQE) et "Perception based Image Quality Evaluator" (PIQE) sont les plus accessibles car ils sont déjà disponibles sur le logiciel Matlab. La figure 2.9 résume le calcul du score d'une image avec un NR-IQA.

Les inconvénients des méthodes NR-IQA ont font des méthodes qui ne sont pas adaptées au calcul de la qualité pour les mosaïques d'images dynamiques. Les NR-IQA retournent un score de qualité qui mesure la distortion de l'image entière. Cette mesure ne répresente pas les artefacts d'une manière optimale. De plus, le modèle de qualité change à chaque fois que l'entraînement est effectué. Ainsi, une image qui a un score optimale avec un modèle de qualité peut avoir un score médiocre avec un modèle de qualité différent. De ce fait, un modèle de qualité fixe est souvent utilisé, ce qui n'est pas optimale dans le cas où une image ne possède pas des images à caractéristiques similaires parmi les images d'entraînements.

Ainsi, la SSIM est la mesure de comparaison qui est le plus souvent utilisée dans l'évaluation des méthodes de création de mosaïques, elle est plus fréquente dans le domaine des mosaïques statiques. Cependant dans le domaine des mosaïques d'images dynamiques, elle peut engendrer des erreurs. Vu

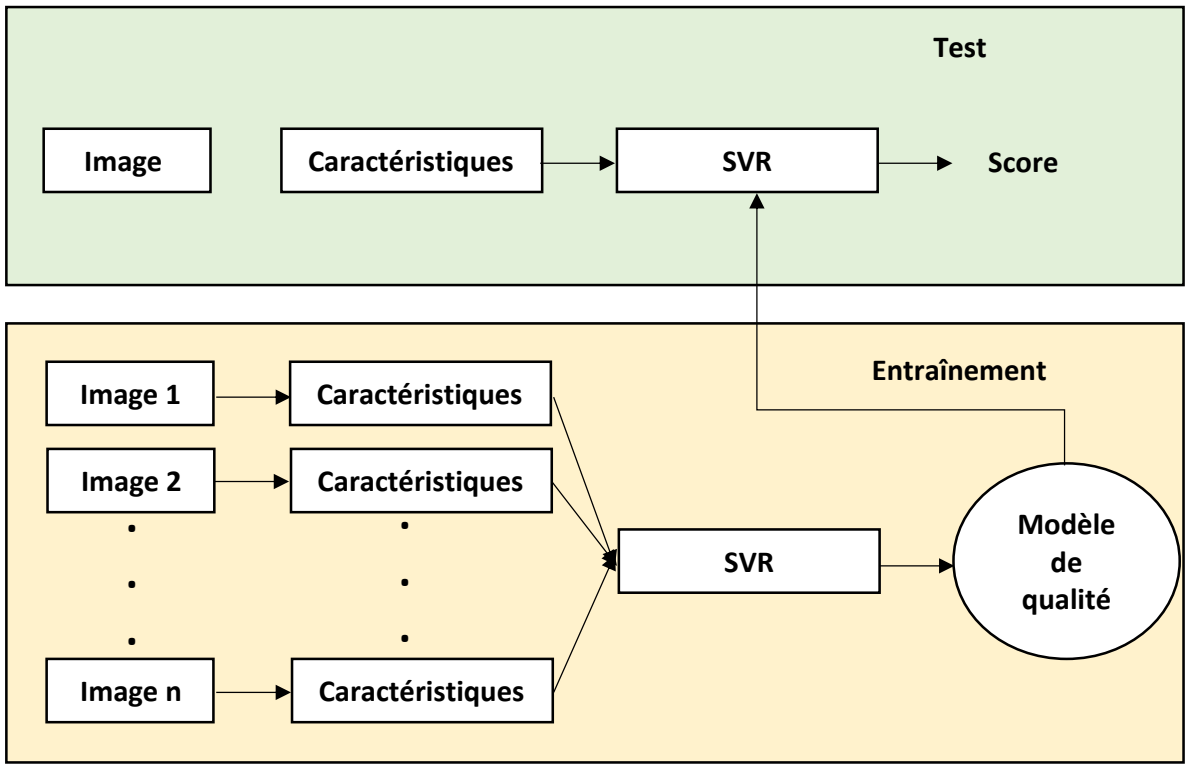


FIGURE 2.9 – Procédure de calcul du score de qualité d'une image avec une méthode NR-IQA

que la comparaison est effectuée sur tous les pixels de la zone de chevauchement, si un artefact est présent sur une partie considérée comme petite par rapport au reste de la zone de chevauchement, la valeur retournée peut être élevée et peut même être optimale dans certains cas malgré la présence d'un artefact. Afin de contourner ce problème, il existe plusieurs moyens qui peuvent être effectués :

- C'est possible d'appliquer un filtre passe haut (Laplacian, Prewitt etc...) sur les images d'entrées et l'image mosaïque pour comparer que les hautes fréquences comme effectuée par Qureshi et al. [66].
- L'une des solutions que nous avons utilisé pour les méthodes basées sur le calcul d'une "seamline" c'est le calcul de la mesure comparaison juste autour des pixels de la "seamline". Effectivement, vu que les mosaïques obtenues par les méthodes qui utilisent les "seamlines" créent des erreurs seulement autour des "seamlines", le meilleur moyen de les comparer c'est de comparer les zones autour des "seamlines". Cependant, cette opération n'est possible que si les positions des pixels de la "seamline" sont connus.
- Une autre solution que nous avons proposé [70], c'est l'utilisation d'un flux optique dense pour déterminer les zones qui contiennent un mouvement entre les images d'entrées et de comparer ces zones extraites avec la mosaïque résultante. Cette comparaison est utile pour toutes les méthodes ou dans le cas où la "seamline" n'est pas connue. De plus, utiliser les champs de mouvement donne une comparaison plus optimale que d'utiliser des blocs pour deux raisons.

Premièrement, si tout le bloc contient une partie d'un artefact quand il sera comparé avec les images d'entrées, il va être considéré comme optimal puisque la valeur optimale entre les images est choisie et deuxièmement, comparer les images que sur les zones dynamiques permet d'enlever l'impact des zones similaires sur la valeur de comparaison finale. L'annexe B présente la méthode en détails.

2.5 Conclusion

Dans ce chapitre, l'étape de traitement des objets dynamiques a été étudiée en détails. Nous avons tout d'abord défini les différents artefacts. Ensuite, nous avons exploré les différentes méthodes de gestion des objets dynamiques et leurs points forts et points faibles. Ces méthodes ont été divisées en deux catégories : la première étant les méthodes basées sur le calcul d'une "seamline" où on a détaillé la méthode de Mills et Dudek [54], la méthode de Zeng et al. [55], la méthode de Yao et al. [37] et la méthode de Li et al. [56]. Puis, la deuxième catégorie sur les méthodes basées sur la reconstruction de régions où la méthode de Uyttendaele et al. [62] et la méthode Bartoli et al. [17] ont été détaillées. Finalement, nous avons présenté la procédure d'évaluation des mosaïques d'images avec différentes mesures de comparaisons.

Chapitre 3

Création de mosaïques d'images dynamiques basées sur un détecteur de contours pour calculer une "seamline" optimale.

3.1 Introduction 50
3.2 Utilisation d'un détecteur de contours pour calculer la "seamline" optimale 52
3.2.1 Prétraitement : Transfert de couleurs 52
3.2.2 Recalage d'images 54
3.2.3 Recherche de la "seamline" avec un détecteur de contours 54
3.2.3.1 Régions Canny 54
3.2.3.2 Carte d'énergie 56
3.2.3.3 Calcul de la "seamline" optimale 57
3.2.4 "Pyramid blending" 59
3.3 Résultats expérimentaux 60
3.3.1 Données synthétiques 60
3.3.2 Données réelles 64
3.3.3 Analyses des mesures obtenues 70
3.4 Conclusion 70

3.1 Introduction

Dans ce chapitre, nous présenterons une nouvelle méthode [71] pour la création de mosaïques d'images dynamiques basée sur un détecteur de contours pour calculer une "seamline" optimale. Cette méthode entre dans la catégorie des méthodes basées sur le calcul d'une "seamline". Par conséquence, elle est séparée en deux étapes : la création de la carte d'énergie et le calcul de la "seamline" à partir de cette carte. Les motivations principales que nous souhaitons réaliser pour cette méthode sont :

- Premièrement, effectuer une meilleur détection des objets dynamiques même avec la présence d'une différence exposure entre les images d'entrées. En effet, une différence d'exposure peut facilement exister même si les images d'entrées ont été capturées au même moment et avec la même caméra en raison de l'effet que la lumière a sur le système optique de la caméra.
- Deuxièmement, créer un algorithme de calcul de "seamline" optimal guidé par des régions définies comme étant sûres et des régions à éviter afin d'avoir un meilleur contrôle sur le résultat pour pouvoir utiliser des images d'entrées avec beaucoup d'objets dynamiques et pour obtenir un bon résultat comparé aux autres méthodes qui utilisent des exemples où les objets dynamiques sont en quantité minime.
- Finalement, obtenir un gain de temps dans la génération de la mosaïque.

Afin de réaliser ses motivations, plusieurs contributions ont été effectuées :

- La première contribution consiste à utiliser un prétraitement qui est le transfert de couleur pour ajuster l'intensité entre les images et ainsi régler la différence d'exposition. L'un des avantages du transfert de couleur c'est le fait qu'il est effectué dans l'espace de couleur CIELab. Le premier canal de cet espace est la luminosité. Par conséquence, régler une différence d'exposure est plus optimale sur ce canal que sur un canal RGB.
- La seconde contribution repose sur l'utilisation de régions créées à partir du détecteur de contours Canny pour générer une carte d'énergie. En outre, la création de ces régions est basée sur le recalage d'images. En effet, en utilisant le détecteur de points d'intérêts SIFT et la mise en correspondance avec RANSAC, deux types de points sont obtenus. Les "inliers", qui sont obtenus par les points d'intérêts similaires et les "outliers", qui sont obtenus par les points d'intérêts non similaires. Par conséquence, en combinant entre les contours détectés avec Canny et la position des "inliers" et des "outliers", c'est possible d'obtenir deux types de régions, les régions "inliers" qui contiennent que des points "inliers" et les régions "outliers" qui contiennent au minimum un point "outlier". Ainsi, en appliquant ces régions sur une carte d'énergie, la détermination des éléments dynamiques par rapport aux éléments statiques est facilement identifiable.
- La dernière contribution est pour le calcul d'une "seamline" optimale en se basant sur les régions "inliers" et les régions "outliers". Vu que les régions "inliers" représentent les régions

communes et les régions "outliers" représentent les zones dynamiques, il suffit de calculer la "seamline" en la guidant pour passer à travers le plus de régions "inliers" et d'éviter un maximum les régions "outliers".

Le plan de ce chapitre est le suivant : nous allons tout d'abord présenter les algorithmes de la création de la mosaïque en détails qui sont le prétraitement avec le transfert de couleurs, le recalage d'images, la création de la carte d'énergie avec la détection des contours et la combinaison entre les régions générés et une différence d'intensité, puis la manière de calculer la "seamline" et le "blending". La figure 3.1 est une représentation graphique de ces différentes étapes pour créer la mosaïque. Finalement, plusieurs mosaïques générées d'images synthétiques et d'images réelles ont été comparées pour évaluer la méthode développée avec d'autres méthodes dans des situations différentes.

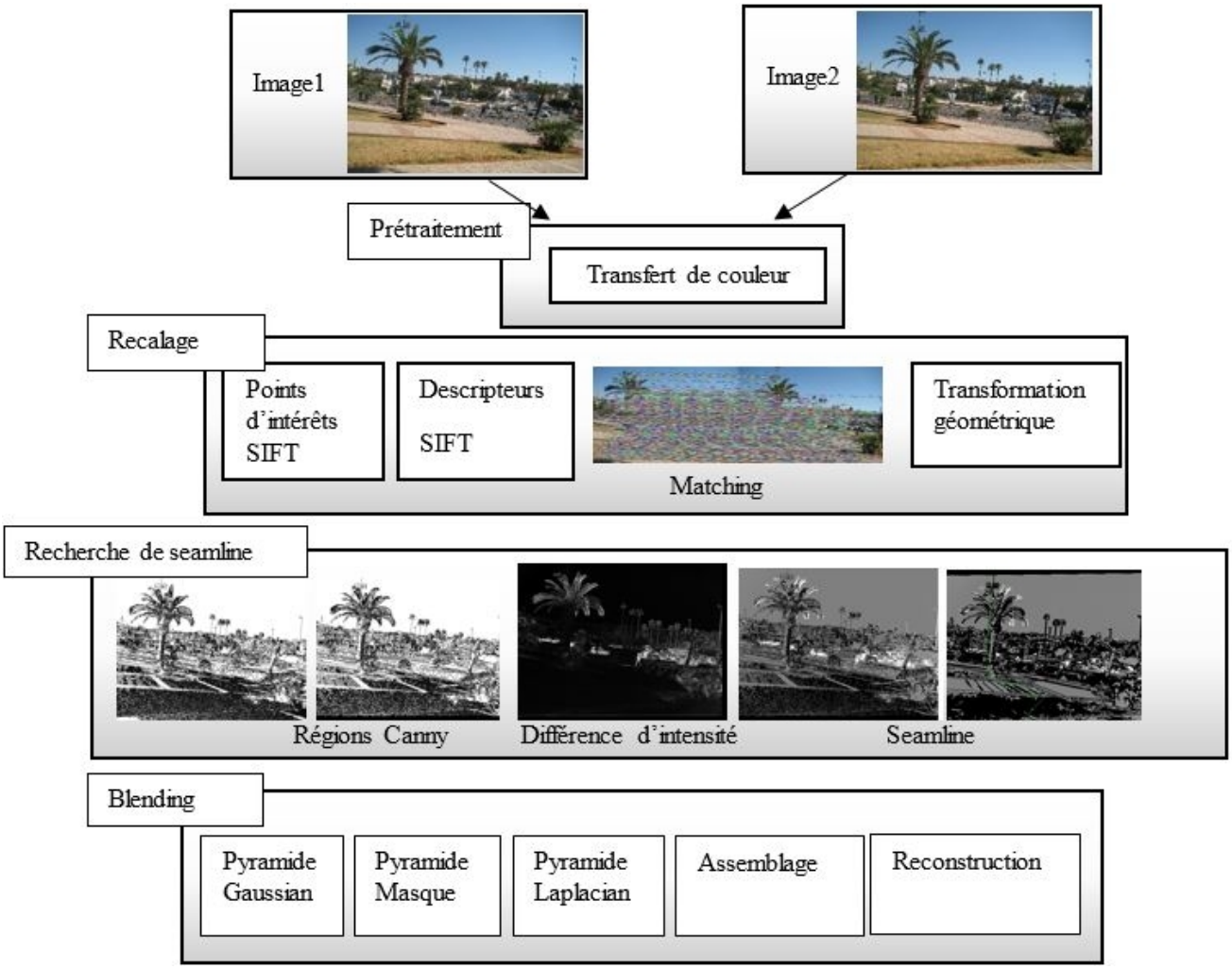


FIGURE 3.1 – Procédure pour la création d'une mosaïque avec la méthode basée sur un détecteur de contours. [71].

3.2 Utilisation d'un détecteur de contours pour calculer la "seamline" optimale

3.2.1 Prétraitement : Transfert de couleurs

C'est rare de trouver des images capturées par des caméras possédant les mêmes intensités de couleurs même s'ils sont capturés par la même caméra. En outre, une différence d'exposition existe toujours aussi minime qu'elle peut être. Un mouvement de la caméra en raison de tremblement de l'utilisateur ou de rotation de la caméra peut créer une différence d'exposition entre les images en raison de la sensibilité des systèmes optiques des caméras par rapport au changement soudain des sources lumineuses de la scène. En effet, un changement simple d'exposition peut avoir des conséquences sur la détection des objets dynamiques dans la carte d'énergie et cet effet a un plus grand impact sur les cartes d'énergie basées sur des différences d'intensités. Afin de corriger ce problème, plusieurs méthodes utilisent une compensation de gain. Cependant, la compensation de gain n'est pas suffisante quand l'exposure est trop importante. Comme précisé dans la section 1.5.1.1, la compensation de gain est calculée après la phase du recalage sur les points "inliers", or dans la présence d'une sous-exposition ou une sur-exposition élevée, le recalage d'images peut échouer. C'est pour cette raison que nous allons utiliser un transfert de couleurs [72] comme prétraitement. Cette méthode est exécutée dans le domaine CIELab qui est un espace de couleurs avec 3 canaux : le canal L qui représente la clarté/luminence (valeurs entre 0 pour noir et 100 pour blanc), le canal $a*$ pour les couleurs allant de vert (valeurs négatives) à rouge (valeurs positives) et le canal $b*$ pour les couleurs de bleu (valeurs négatives) à jaune (valeurs positives).

Le principe du transfert de couleurs est de calculer la moyenne et l'écart type de chaque canal de l'espace couleur CIELab pour chaque image puis de transformer les couleurs d'une image pour qu'elle ressemble à l'image de référence.

Soit Im_1 et Im_2 deux images. Le prétraitement est effectué de la façon suivante :

- Transformer les images Im_1 et Im_2 du canal RGB au canal CIELab.
- Calculer la moyenne $\mu_{1,k}$ et $\mu_{2,k}$ et l'écart type $\sigma_{1,k}$ et $\sigma_{2,k}$ des images Im_1 et Im_2 respectivement pour chaque canal $k = L, a*, b*$.
- Calculer les nouvelles valeurs des pixels (x, y) de l'image Im_1 pour obtenir l'image $Im_1^{'}$ par l'équation 3.1.

$$Im_{1,k}^{'}(x,y) = (Im_{1,k}(x,y) - \mu_{1,k}) \times \frac{\sigma_{1,k}}{\sigma_{2,k}} + \mu_{2,k} \tag{3.1}$$

  Avec k est un canal tel que $k = L, a*, b*$.
- Transformer l'image $Im_1^{'}$ du canal CIELab au canal RGB.

Le but principal de ce prétraitement c'est d'ajuster les intensités entre les images pour une meilleure détection des objets dynamiques et ainsi une meilleure mosaïque. Afin de tester l'effet du

transfert de couleur, nous avons comparé le résultat obtenu entre une image de référence (Figure 3.2.a) et la même image sous différentes exposures (Figure 3.2.b) de la base de données "Oxford standard data set" [74]. Cette base de données contient plusieurs ensembles d'images utilisées dans des tests de création de descripteurs. L'ensemble d'images utilisé est l'ensemble "Leuven". Le résultat obtenu est meilleur dans la figure 3.2.c.



(a) Première image de l'ensemble et image de référence



(b) Les images 2 à 6 de l'ensemble avant le transfert de couleur



(c) Les images 2 à 6 de l'ensemble après le prétraitement

FIGURE 3.2 – Application du transfert de couleur sur l'ensemble d'images Leuven de la base de données "Oxford standard data set".

Pour montrer l'efficacité du transfert de couleur, la mesure de similarité de couleur "Peak to Signal Noise Ratio" (PSNR) [75] est calculée (Equation 3.2). PSNR est un ratio entre la valeur la plus élevée dans un signal (ou image) et un bruit. Quand le bruit est peu présent (Dans ce cas le bruit c'est l'exposition), la valeur de PSNR est élevée et la qualité de l'image est bonne.

$$PSNR(I_r, I_k) = 10 \times log_{10}\left(\frac{max(I_r)^2}{\frac{1}{n \times m}\sum_{x=0}^{n-1}\sum_{y=0}^{m-1}(I_r(x,y) - I_k(x,y))^2}\right) \tag{3.2}$$

Où :

- I_r est l'image de référence.
- I_k est l'image que nous souhaitons comparer.
- n le nombre de lignes et m le nombre de colonnes.
- $max(I_r)$ la valeur d'intensité la plus élevée dans l'image I_r.

Le tableau 3.1 montre les valeurs PSNR obtenues en comparant l'image de référence de la figure 3.2 avec les autres images de l'ensemble d'images Leuven avant le transfert de couleur et après. Les résultats obtenus montrent une amélioration de la qualité de l'image, en moyenne une amélioration de la qualité des images de **+42,55%** a été réalisée.

	Img2	Img3	Img4	Img5	Img6
Avant le prétraitement	21,2705	19,5750	16,7241	14,4845	12,8650
Après le prétraitement	26,5215	25,2393	23,8405	21,9952	21,1932
Amélioration de	+24,68%	+28,93%	+42,55%	+51,85%	+64,73%
Moyenne de : **+42,55%**					

TABLE 3.1 – Valeurs PSNR obtenues entre l'image de référence (Figure 3.2.a) et les images avant le transfert de couleurs (Figure 3.2.b) et après le transfert de couleurs (Figure 3.2.c)

3.2.2 Recalage d'images

Le recalage d'image utilisé dans cette méthode combine entre le détecteur de points d'intérêts SIFT [24] (Section 1.3.4.1) et la mise en correspondance avec RANSAC [38] (Section 1.3.4.2). Vu que la méthode dévelopée est basée sur l'utilisation des "inliers" et "outliers", ces points sont tous stockés pour les utiliser plus tard dans la création des régions "inliers" et régions "outliers".

3.2.3 Recherche de la "seamline" avec un détecteur de contours

Etant donné que le but final de cette méthode c'est le calcul d'une "seamline" optimale pour la création d'une mosaïque sans artefact. Dans cette sous-section, nous allons présenter les étapes pour la recherche de la "seamline". Tout d'abord, des régions que nous avons nommé des régions "inliers" et régions "outliers" vont être définies avec le détecteur de contours Canny [73] et les "inliers" et "outliers" obtenus du recalage d'images. Ensuite une carte d'énergie est calculée en combinant entre une différence d'intensité et les régions de Canny. En effet, cette carte permet de surligner facilement les régions qui peuvent créer des artefacts. Finalement, un algorithme de recherche de "seamline" est présenté. Ce dernier est programmé de la façon qu'il évite les régions "outliers" et passe par les régions "inliers".

3.2.3.1 Régions Canny

Dans cette étape, la création de régions que nous catégoriserons en deux types de régions, les régions "inliers" et les régions "outliers", est effectuée.

Dans le but de trouver ces régions d'une manière précise, nous avons utilisé le détecteur de contours Canny [73]. Tout d'abord, le détecteur est utilisé pour obtenir les contours des images d'entrées (avant la phase du recalage), ces contours vont définir plusieurs régions dans l'image. Ensuite, un algorithme de remplissage "$Floodfill$" [76] est utilisé sur chaque région obtenue qui contient un pixel "inlier" ou un pixel "outlier". En effet, l'algorithme "$Floodfill$" affecte à tous les pixels d'une région la même valeur, ainsi c'est possible d'attribuer aux régions d'"inliers" et "outliers" une valeur précise. Pour les régions "outliers", une valeur de 1 (blanc) a été attribuée tandis qu'une valeur de 0.1 a été attribuée aux régions "inliers". Ces valeurs sont facultatives et n'ont aucun impact à part pour distinguer si une région est une région "outlier" ou "inlier". Les régions avec une valeur de 0 sont des régions qui ne contiennent aucun "outlier" ou "inlier".

Soit Im_1 et Im_2 (Figure 3.3.a) deux images. Les régions Canny sont obtenues de la façon suivante :

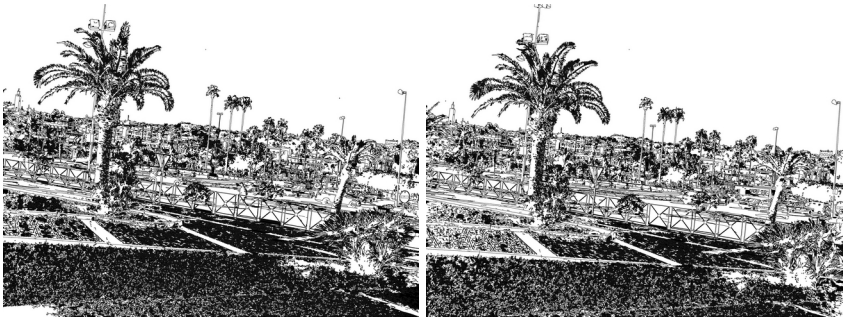
- Application d'un filtre Gaussian $G(x, y, \delta)$ sur les images Im_1 et Im_2 pour enlever le bruit et obtenir les images $Im_1^{'}$ et $Im_2^{'}$.
- Trouver les contours $Mask_1$ et $Mask_2$ des images $Im_1^{'}$ et $Im_2^{'}$ en appliquant un filtre Canny.
- Utiliser une dilatation pour fermer les trous qui peuvent se trouver dans les contours de $Mask_1$ et $Mask_2$. (Figure 3.3.b)
- Pour chaque point "inlier" et "outlier" (Figure 3.3.a), appliquer l'algorithme de remplissage "$Floodfill$" sur chaque région de $Mask_1$ et $Mask_2$ tel que si une région contient au minimum un point "outlier" une valeur de 1 est affectée à la région et si une région contient que des points "inliers" une valeur de 0.1 est assignée. Ceci permettra de créer les régions Canny M_1 et M_2 (Figure 3.3.c).
- Appliquer la transformation géométrique calculée dans la phase de recalage pour mettre M_1 et M_2 dans le même repère.



(a) Images d'entrées avec la position des points "inliers" en vert et points "outliers" en rouge



(b) $Mask_1$ et $Mask_2$ après dilatation



(c) Régions Canny

FIGURE 3.3 – Procédure de créations des régions Canny.

Cette étape a été effectuée avant la transformation géométrique pour éviter les erreurs de précisions.

3.2.3.2 Carte d'énergie

Afin de créer la carte d'énergie, nous avons effectué une différence d'intensité. En effet, cette différence permet d'assigner des valeurs élevées aux objets dynamiques et des valeurs basses aux zones communes. Elle est calculée par l'équation 3.3.

$$\delta_{xy}^{Im} = \frac{abs(I'_{1xy} - I'_{2xy})}{max(I'_{1xy}, I'_{2xy})} \tag{3.3}$$

Où I'_{1xy} et I'_{2xy} sont les zones de chevauchements des images Im_1 Im_2 après application d'un filtre Gaussian. La valeur retournée est entre 0 et 1.

Au même moment de la création de la différence d'intensité, la zone de chevauchement des régions Canny M_1 et M_2 (après la transformation géométrique) des images Im_1 et Im_2 sont fusionnées pour obtenir un masque M qui contient toutes les informations sur les régions "outliers" et "inliers" en une seule image. En effet, si un pixel appartient à une région "outlier", alors dans la fusion il fera partie d'une région "outlier". Cependant, pour un pixel qui appartient à une région "inlier", il a besoin d'appartenir à une région "inlier" dans les deux images pour qu'il soit considéré comme faisant partie d'une région "inlier" dans la fusion.

Après l'obtention de la différence d'intensité δ_{xy}^{Im} et le masque Canny M, les deux sont combinés et des seuils sont appliqués pour obtenir la carte d'énergie finale. Trois types de régions pourront être observées dans la carte d'énergie : les régions des objets dynamiques qui peuvent créer des artefacts, les régions communes, et les régions inchangées. De plus, c'est possible d'extraire l'origine des objets dynamiques, en utilisant un tableau *sign* qui enregistre le signe de la différence d'intensité de l'équation 3.3 avant d'appliquer la valeur absolue. La procédure est la suivante :

- Calcul de la différence d'intensité par l'équation 3.3. (Figure 3.4.a)
- Création du masque M à partir de la zone de chevauchement des images M_1 et M_2 par l'équation 3.4 (Figure 3.4.b)

$$M_{xy} = \begin{cases} 1 & M_{1xy} = 1 \;\; || \;\; M_{2xy} = 1 \\ 0.1 & M_{1xy} = 0.1 \;\; \&\& \;\; M_{2xy} = 0.1 \\ 0 & other \end{cases} \tag{3.4}$$

- Combinaison du masque Canny M et de la différence d'intensité δ_{xy}^{Im} pour obtenir $\delta_{xy}^{Im'}$ par l'équation 4.5.

$$\delta_{xy}^{Im'} = \begin{cases} 0.5 * M_{xy} + 0.5 * \delta_{xy}^{Im} & M_{xy} = 1 \\ 0.5 * \delta_{xy}^{Im} & M_{xy} = 0.1 \\ \delta_{xy}^{Im} & M_{xy} = 0 \end{cases} \tag{3.5}$$

- Application de deux seuils (Equation 3.6) pour obtenir la carte d'énergie finale (Figure 3.4.c) où le premier seuil $Thrd1 = 0.7$ est pour les régions dynamiques et le deuxième seuil $Thrd2 = 0.1$ pour les zones communes.

$$\delta_{xy}^{Im'} = \begin{cases} 1 & \delta_{xy}^{Im'} >= Thrd1 \\ 0 & \delta_{xy}^{Im'} <= Thrd2 \\ \delta_{xy}^{Im'} & else \end{cases} \tag{3.6}$$



(a) Différence d'intensité (b) Fusion des régions Canny (c) Carte d'énergie Canny

FIGURE 3.4 – Procédure d'obtention de la carte d'énergie finale.

3.2.3.3 Calcul de la "seamline" optimale

Une "seamline" optimale est une "seamline" qui va séparer les deux images d'entrées sans créer de discontinuités ou couper des objets dynamiques. Dans le but d'obtenir une "seamline" optimale, il est nécessaire de trouver le chemin qui va passer par le plus de régions "inliers" et qui va éviter toutes les régions "outliers" puisque les régions "inliers" représentent les zones communes et les régions "outliers" n'existent que dans une seule image.

Dans l'étape précédente, la carte d'énergie qui contient les régions "inliers" et "outliers" a été calculée. Effectivement, grâce à cette carte c'est possible de choisir le point de départ de la "seamline" plus facilement, d'éviter les zones contenant des objets dynamiques et de traverser les zones similaires. Ainsi, en utilisant cette carte d'énergie, voici les étapes pour le calcul de la "seamline" optimale dynamiquement :

- Premièrement, les points ayant la valeur la plus petite dans la carte d'énergie $\delta^{Im'}$ sont choisis et stockés dans une liste *minPoints*. Ces points vont être le début de plusieurs "seamlines".
- Deuxièmement, pour chaque point de la liste *minPoints*, une boucle est effectuée sur le point courant pour choisir le point suivant parmi les 5 points adjacents du point initial (Figure 3.5) tel que le point suivant, qui deviendra le point courant à la prochaine itération, est le point ayant la valeur la plus petite parmi les 5 points. La boucle se termine quand un point fait partie de la dernière ligne de $\delta^{Im'}$. Dans cette deuxième étape plusieurs "seamlines" vont être créées. Dans le cas d'effet de bord, les pixels hors bordures ne sont pas prises en compte.
- Finalement, puisque chaque "seamline" n'aura pas le même nombre de points, un pourcentage des points ayant la valeur 1 parmi tous les points est calculé. La "seamline" qui a le pourcen-

tage le plus bas est choisi comme "seamline" optimale. Si plusieurs "seamline" ont le même pourcentage des valeurs 1, un pourcentage des valeurs 0 est calculé entre ces "seamlines" et la "seamline" avec le pourcentage le plus élevé est choisi.

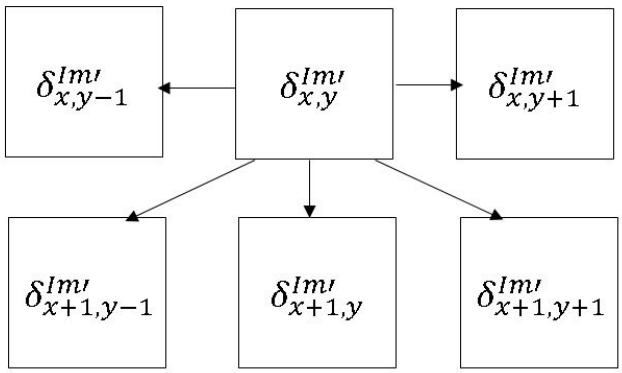
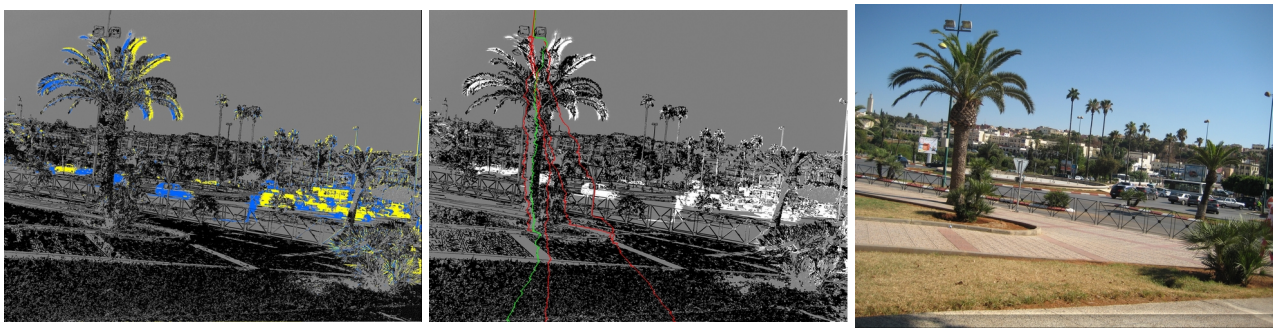


FIGURE 3.5 – Choix des points adjacents pour le calcul de la "seamline".

C'est probable qu'un voisinage d'un pixel contient des pixels avec valeurs similaires, c'est pour ça qu'un ordre pour le choix des points (Figure 3.5) est choisi. Cependant, même avec cette précaution, dans la présence d'un nombre élevé d'éléments dynamiques, la "seamline" va inévitablement passer par un objet dynamique. Pour cette raison, le tableau *sign* précédemment calculé peut être utilisé. En effet, dans ce tableau les pixels auront des valeurs positives ou négatives, où les pixels des objets dynamiques d'une valeur positive ou négative auront tendances à appartenir à la même image. Ainsi dans ce cas, il est préférable d'avoir la "seamline" passer à droite dans le cas de valeurs positives et à gauche quand les valeurs sont négatives.

La figure 3.6.a présente un exemple de la position des éléments dynamiques avec des valeurs positives et négatives et la figure 3.6.b montre les différentes "seamlines" calculées avec la meilleure "seamline" qui est en vert.



(a) (b) (c)

FIGURE 3.6 – Calcul de la "seamline" et mosaïque finale, (a) Tableau *sign* appliqué sur $\delta^{Im'}$ avec les valeurs positives en bleu et négatives en jaune, (b) "Seamline" choisie en vert parmi les différentes "seamlines" calculées en rouge, (c) Mosaïque finale après le "blending".

3.2.4 "Pyramid blending"

L'étape finale est l'étape du "blending". Le "blending" effectué dans cette méthode est le "pyramid blending" [43] (Section 1.4.1.3). La figure 3.6.c montre la mosaïque finale obtenue.

La figure 3.7 résume les trois étapes de la méthode développée dans ce chapitre.

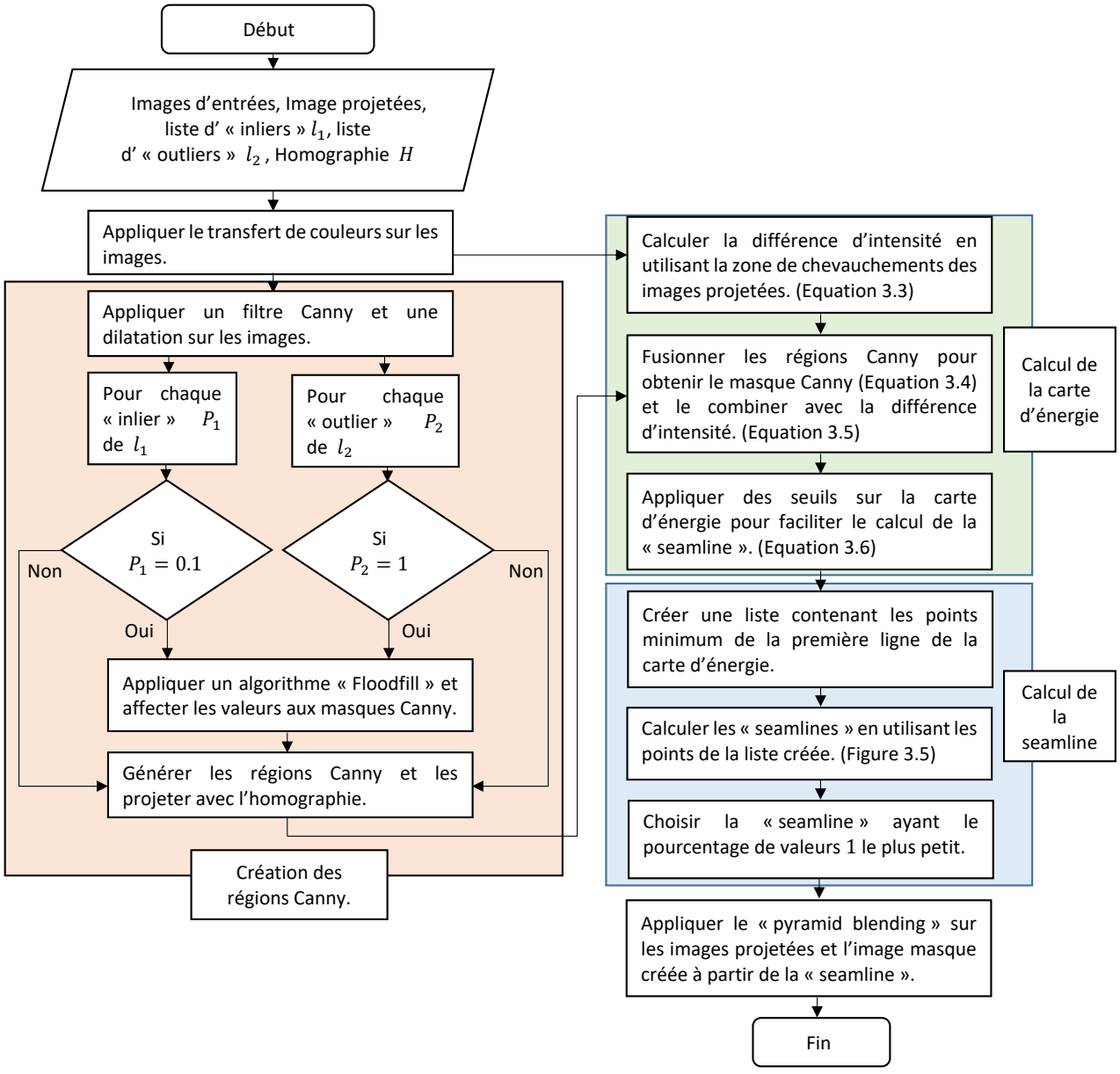


FIGURE 3.7 – Algorithme de la méthode développée basée sur le détecteur de contours de Canny pour le calcul d'une "seamline" optimale et la création d'une mosaïque.

3.3 Résultats expérimentaux

Deux ensembles d'images ont été utilisés pour montrer l'efficacité de la méthode basée sur le détecteur de contours Canny pour calculer une "seamline" optimale. Le premier ensemble contient des données synthétiques tirées de la base de données DAVIS [77], tandis que le deuxième ensemble contient des images réelles capturées avec une caméra digitale Canon Digital IXUS 60. Chacun de ces ensembles contient cinq paires d'images variées avec un type de mouvement. Pour chaque paire, une mosaïque a été créée avec la méthode de Mills et Dudek [54] (Section 2.3.1.1), la méthode CSDP de Zeng et al. [55] (Section 2.3.1.2) et la méthode basée sur le détecteur Canny. Puis la comparaison de ces méthodes a été effectuée sous trois critères : le chemin de la "seamline", la qualité de la mosaïque et le temps d'exécution.

La comparaison de la qualité de la mosaïque a été réalisée avec les mesures de comparaisons suivantes :

- La "Structure Similarity Index Measure" (SSIM) [65] (Section 2.4.2.1) pour la comparaison géométrique et structurelle.
- Le "Spectral Angle Mapper" (SAM) [68] (Section 2.4.2.3) pour la comparaison de la qualité des couleurs.
- Le "Intensity Magnitude Ratio" (IMR) [69] (Section 2.4.2.4) pour la comparaison des valeurs d'intensité.

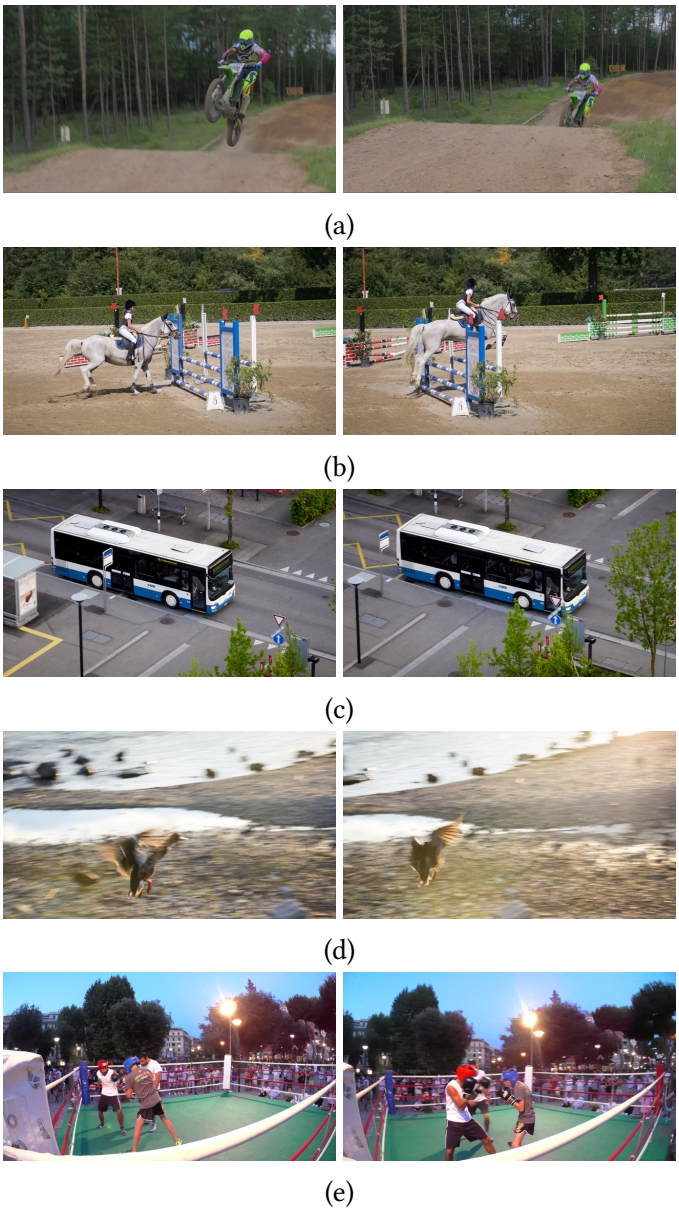
De plus, la méthode SSIM a été calculée avec la méthode de Qureshi et al. [66] où un filtre passe haut est utilisé pour comparer que les hautes fréquences des images.

Les différentes méthodes ont été codées en Java avec la librairie OpenCV avec un PC portable dont le microprocesseur est un Intel Core i7-3632QM CPU à 2.20 GHz.

3.3.1 Données synthétiques

La figure 3.8 montrent les paires d'images utilisées. Ces images font partie de la base de données DAVIS [77] qui est une base de données utilisée pour la segmentation d'objets dynamiques. Chaque paire d'images a été choisie pour tester différentes situations :

- La première paire d'images 3.8.a contient un objet en mouvement dans une grande zone de chevauchement.
- La seconde paire 3.8.b montre un mouvement effectué sur la moitié de la zone de chevauchement.
- Différemment de la deuxième paire, la troisième paire 3.8.c contient un mouvement qui est dans toute la zone de chevauchement.
- Ensuite, la quatrième paire 3.8.d contient une différence d'exposure
- Et finalement, la cinquième paire 3.8.e est une paire où il y a beaucoup d'objets dynamiques.

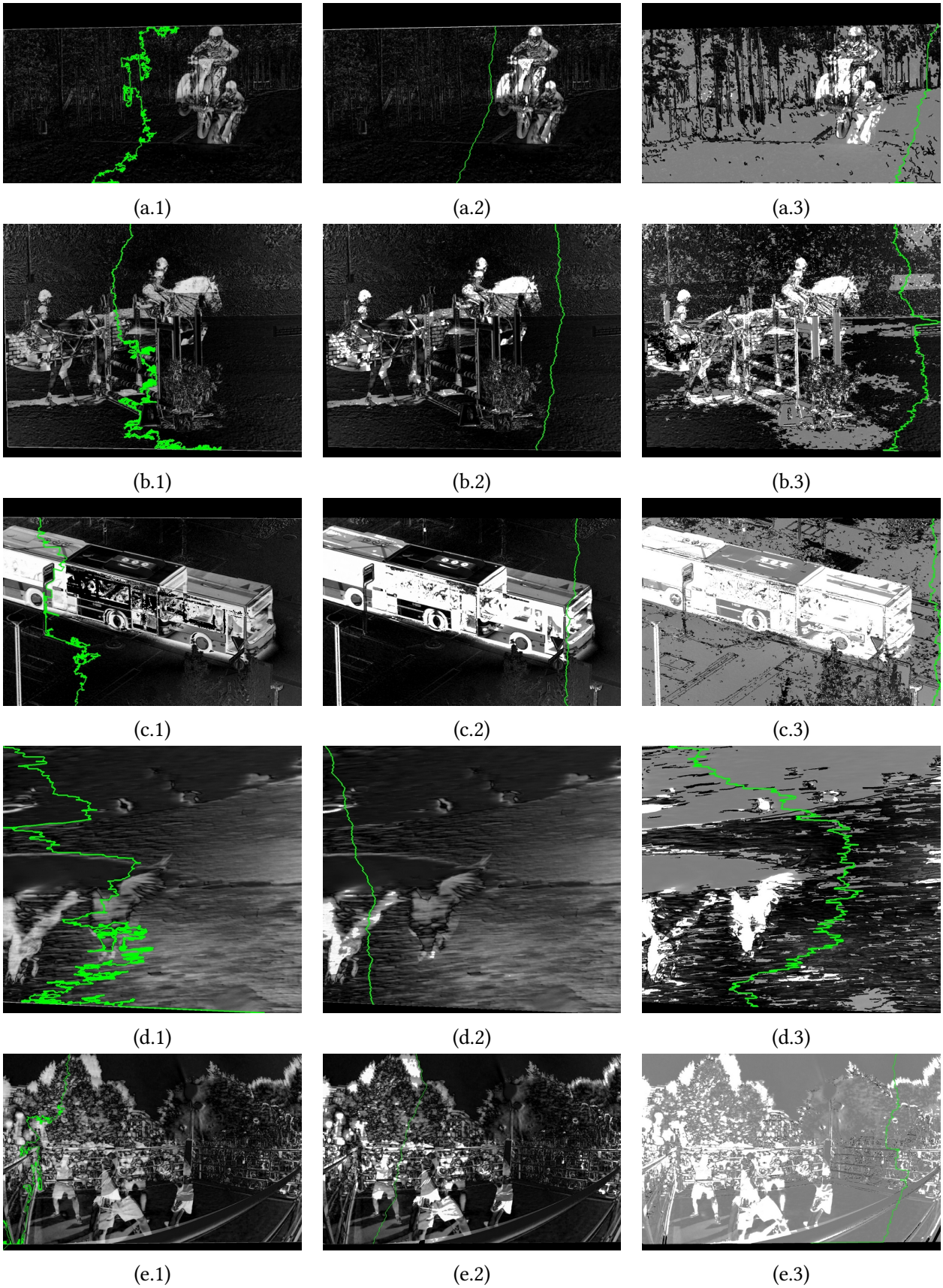


(a) (b) (c) (d) (e)

FIGURE 3.8 – Paires d'images de la base de données DAVIS (a) Motocross-jump (b) Horsejump-low (c) Bus (d) Mallard-fly (e) Boxing-fisheye.

Après la projection des paires d'images dans le même système de coordonnées, la prochaine étape est le calcul de la "seamline". Chaque méthode a calculé sa "seamline" avec sa propre carte d'énergie et les résultats des cartes d'énergie et des "seamlines" sonts présentées dans la figure 3.9. Dans l'ensemble, la méthode développée possède une meilleur détection des objets dynamiques dans la carte d'énergie, cette observation est visible sur les différentes "seamlines" de la figure 3.9. En effet, pour la première paire (Figure 3.8.a), aucune des "seamlines" n'as coupé des objets dynamiques (Figure 3.9.a) ce qui a permis d'obtenir de bons résultats pour les mosaïques (Figure 3.10.a). Pour la seconde paire (Figure 3.8.b), seulement la méthode Mills (Figure 3.9.a.1) a traversé un objet dynamique. Cependant, pour les paires restantes (Figure 3.8.c, 3.8.d, 3.8.e), seulement la méthode développée a réussi à éviter tous les objets dynamiques. Les artefacts qui sont apparus dans la figure 3.9 sont encadrés en rouge sur la figure 3.10 qui montre les mosaïques obtenues avec les "seamlines" calculées par les différentes

méthodes.



(a.1) (a.2) (a.3)

(b.1) (b.2) (b.3)

(c.1) (c.2) (c.3)

(d.1) (d.2) (d.3)

(e.1) (e.2) (e.3)

FIGURE 3.9 – "Seamlines" des images de la base de données DAVIS (a) Motocross-jump (b) Horsejump-low (c) Bus (d) Mallard-fly (e) Boxing-fisheye, (1) Mills (2) CSDP (3) Notre méthode.

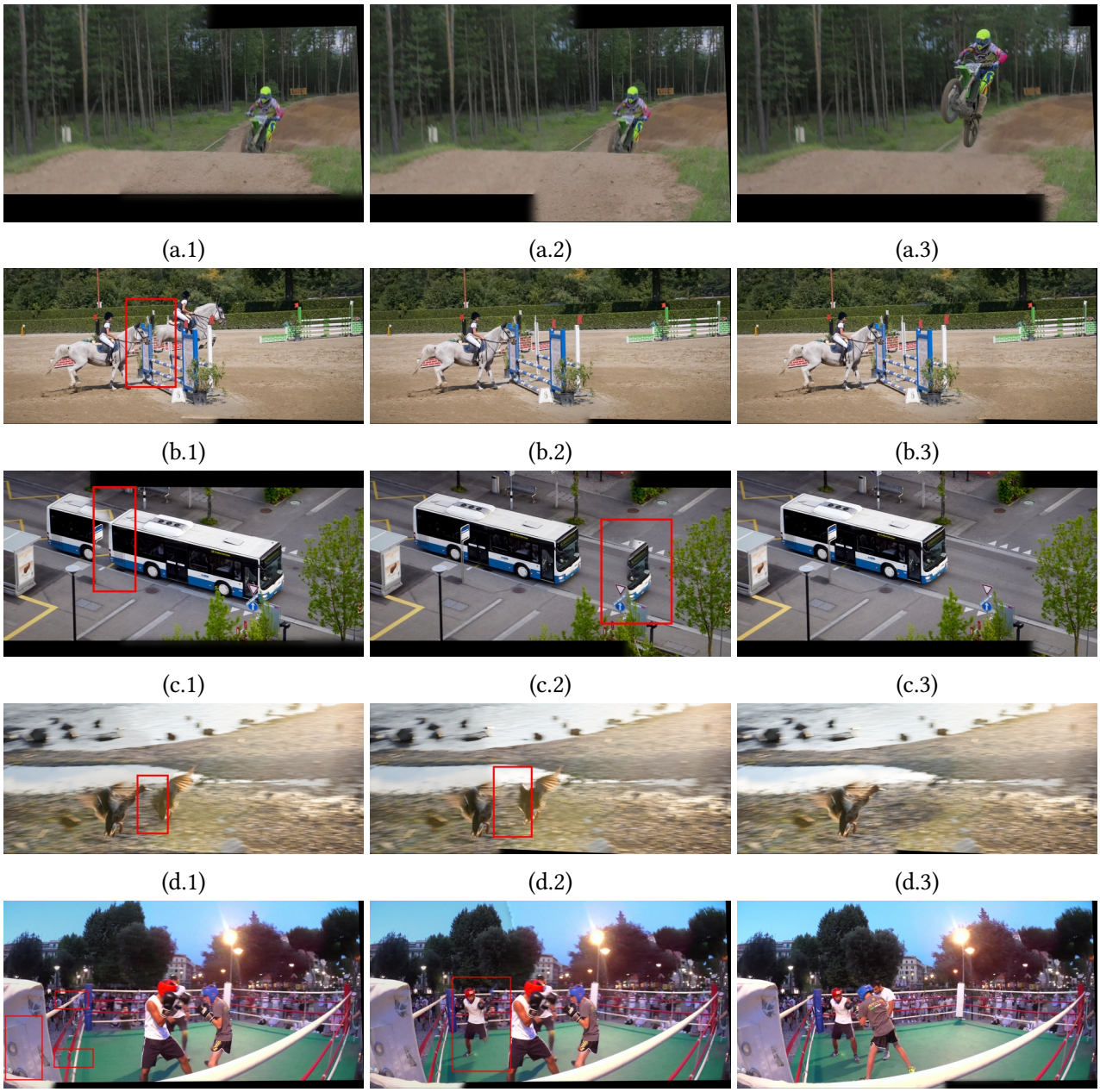


FIGURE 3.10 – Mosaïques des images de la figure 3.8 (a) Motocross-jump (b) Horsejump-low (c) Bus (d) Mallard-fly (e) Boxing-fisheye, (1) Mills (2) CSDP (3) Notre méthode.

Les valeurs de comparaison des différentes méthodes (Mills, CSDP et la méthode développée) obtenues avec la "Structure Similarity Index Measure" (SSIM), le "Spectral Angle Mapper" (SAM) et l'"Intensity Magnitude Ratio" (IMR) sont affichées dans le tableau 3.2. En analysant cette table, nous avons remarqué que la méthode développée produit des mosaïques avec de meilleures qualités. En effet, presque toutes les valeurs de la comparaison de la qualité des couleurs (SAM) sont inférieures à 2 ce qui représente des valeurs optimales. De plus, les mesures de la qualité d'intensité (IMR) de notre méthode sont supérieures aux autres méthodes. Tandis que pour les mesures géométriques (SSIM) toutes les valeurs sont optimales. (Les valeurs optimales sont en gras)

	SSIM			SAM			IMR		
	Mills	CSDP	Notre méthode	Mills	CSDP	Notre méthode	Mills	CSDP	Notre méthode
(a)	0,9759	**0,9804**	**0,9828**	4,9224	4,2493	**0,7737**	**0,9067**	0,9039	0,9908
(b)	**0,9959**	**0,9981**	**0,9979**	6,0155	2,7719	**0,4586**	**0,9080**	0,9517	0,9957
(c)	**0,9938**	**0,9929**	**0,9976**	8,8541	2,9010	**0,1939**	0,7776	0,8644	0,9563
(d)	**0.9887**	**0.9961**	**0.9943**	4.3496	4.3456	3.4808	0.5796	0.4860	0.8336
(e)	**0.9979**	**0.9966**	**0.9984**	11.3401	9.0546	**1.1243**	0.8838	0.8919	0.9721

TABLE 3.2 – Valeurs SSIM, SAM et IMR pour les mosaïques créées à partir de la figure 3.8

Finalement, le dernier critère est la comparaison des temps d'exécution. La figure 3.11 affiche la comparaison entre les temps d'exécution des différentes méthodes. La méthode développée est légèrement plus rapide que les autres méthodes. A noter, que plus les images sont plus grands, plus la différence en terme de rapidité est plus grande dans l'obtention de la mosaïque (Les paires de la figure 3.8.a, 3.8.b, 3.8.c et 3.8.d ont une taille de $(1920, 1080)$, tandis que la figure 3.8.e a une taille de $(3840, 2160)$).

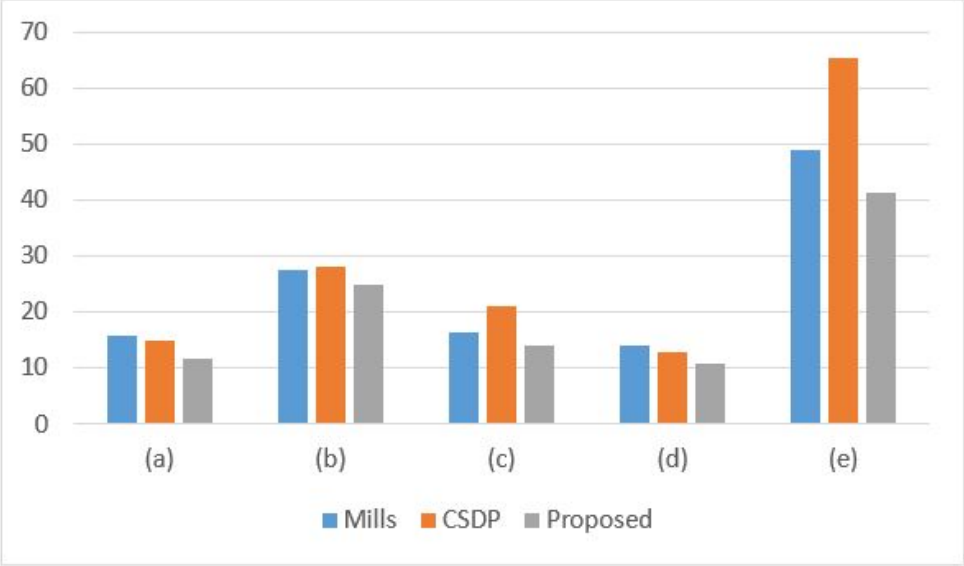


FIGURE 3.11 – Temps d'exécution des méthodes Mills, CSDP et notre méthode pour les images de la figure 3.8.

3.3.2 Données réelles

Les données présentées dans la figure 3.12 sont des images réelles capturées avec une caméra digitale Canon Digital IXUS 60.

Les trois premières paires (3.12.a, 3.12.b et 3.12.c) contiennent un nombre bas d'objets dynamiques tandis que 3.12.d et 3.12.e possèdent un nombre plus élevé. En plus, les zones de chevauchements des paires sont variées. En effet, quand la zone de chevauchement est petite, obtenir un bon résultat devient plus difficile. Nous notons aussi que la dernière paire 3.12.e possède une différence d'exposure où la seconde image est sous-exposée et que la troisième paire 3.12.c est légèrement sur-exposée.



(a)



(b)



(c)



(d)



(e)

FIGURE 3.12 – Paires d'images capturées (a) People (b) Valley (c) Cars (d) Plaza (e) Door.

Le prétraitement avec le transfert de couleurs est utilisé pour enlever l'exposure entre les images. La figure 3.13 montre les images avant et après le prétraitement. Le résultat du prétraitement peut aussi être observé sur le tableau 3.3 qui montre les différentes valeurs de la mesure de similarité des couleurs PSNR. En effet, nous pouvons remarquer une amélioration où les valeurs de PSNR sont plus élevées après le prétraitement.



(a)

(b)

(c)

FIGURE 3.13 – Séries des données réelles de la figure 3.12 (a) Image de référence (b) Seconde image avant le prétraitement (c) Résultat après le prétraitement.

	(a)	(b)	(c)	(d)	(e)
Avant le prétraitement	19,5615	16,0830	13,6095	19,2807	15,0467
Après le prétraitement	19,9177	16,2564	14,0219	20,9518	18,5043

TABLE 3.3 – Valeurs PSNR obtenues entre les paires des images réelles de la figure 3.12)

Après le prétraitement, les "inliers" et "outliers" ont été calculés par SIFT et la mise en correspondance a été effectuée par RANSAC puis les images ont été projetées dans le même repère avec l'homographie. Le résultat des cartes d'énergie et des "seamlines" de chaque méthode est visible dans la figure 3.14. En effet, dans cette figure, c'est visible que pour la méthode Mills et la méthode CSDP, la "seamline" est passée par plusieurs objets dynamiques tandis que la méthode développée les a évité. Ce résultat est notamment grâce aux régions "outliers" qui rendent les pixels pouvant créer des artefacts plus visibles par rapport aux autres méthodes. En combinant ces régions avec les régions "inliers", le calcul d'une "seamline" optimale devient plus simple. Par exemple, dans la figure 3.14.c.3 les arbres à droite de l'image font partie des régions "inliers" ainsi leur valeurs dans la carte d'énergie est basse. Cependant pour les autres méthodes 3.14.c.1 et 3.14.c.2, l'intensité des arbres est élevée ce qui a causé l'apparition de "ghosting" puisque le chemin avec les "inliers" a été considéré comme un mauvais chemin.

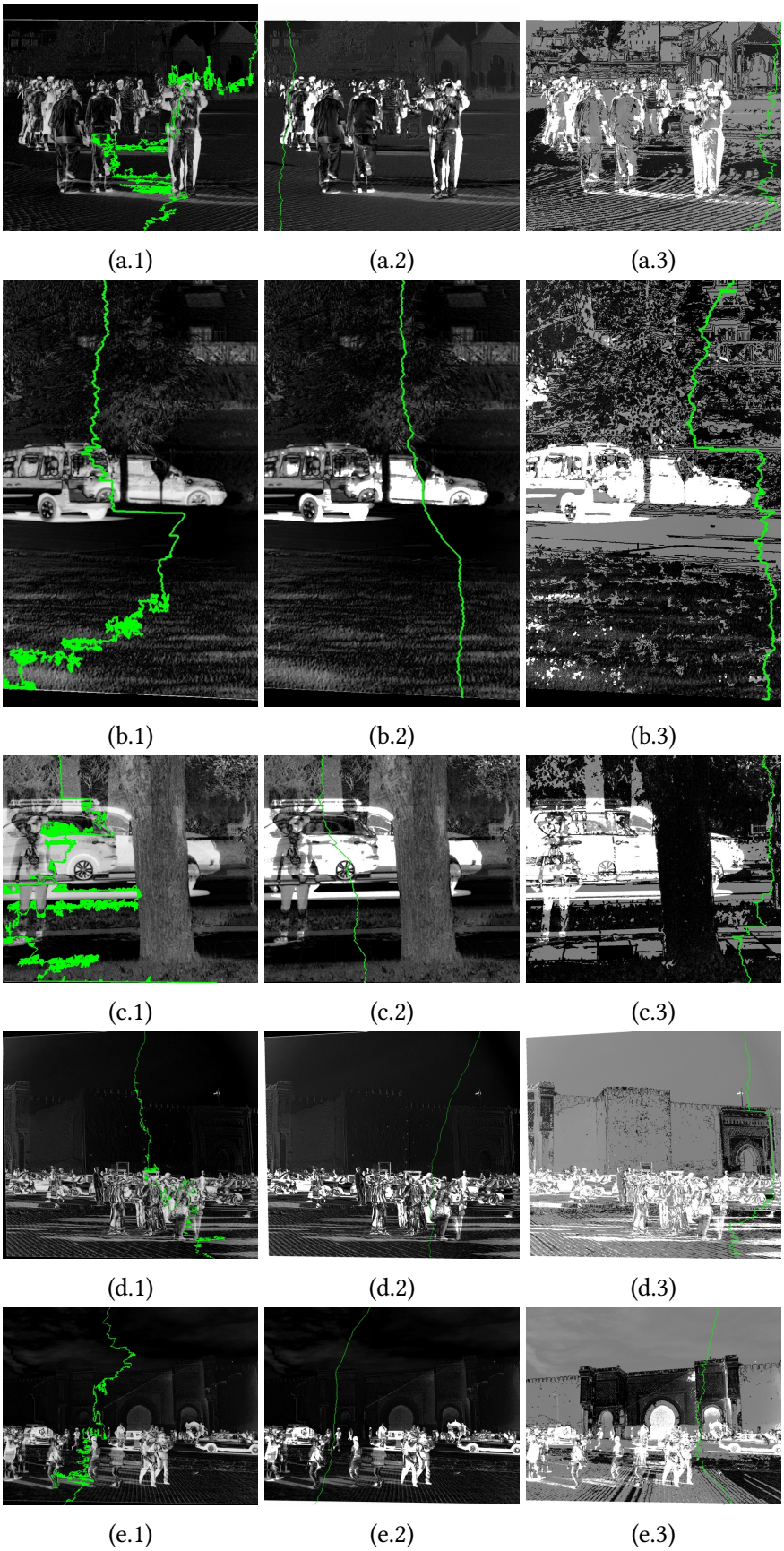
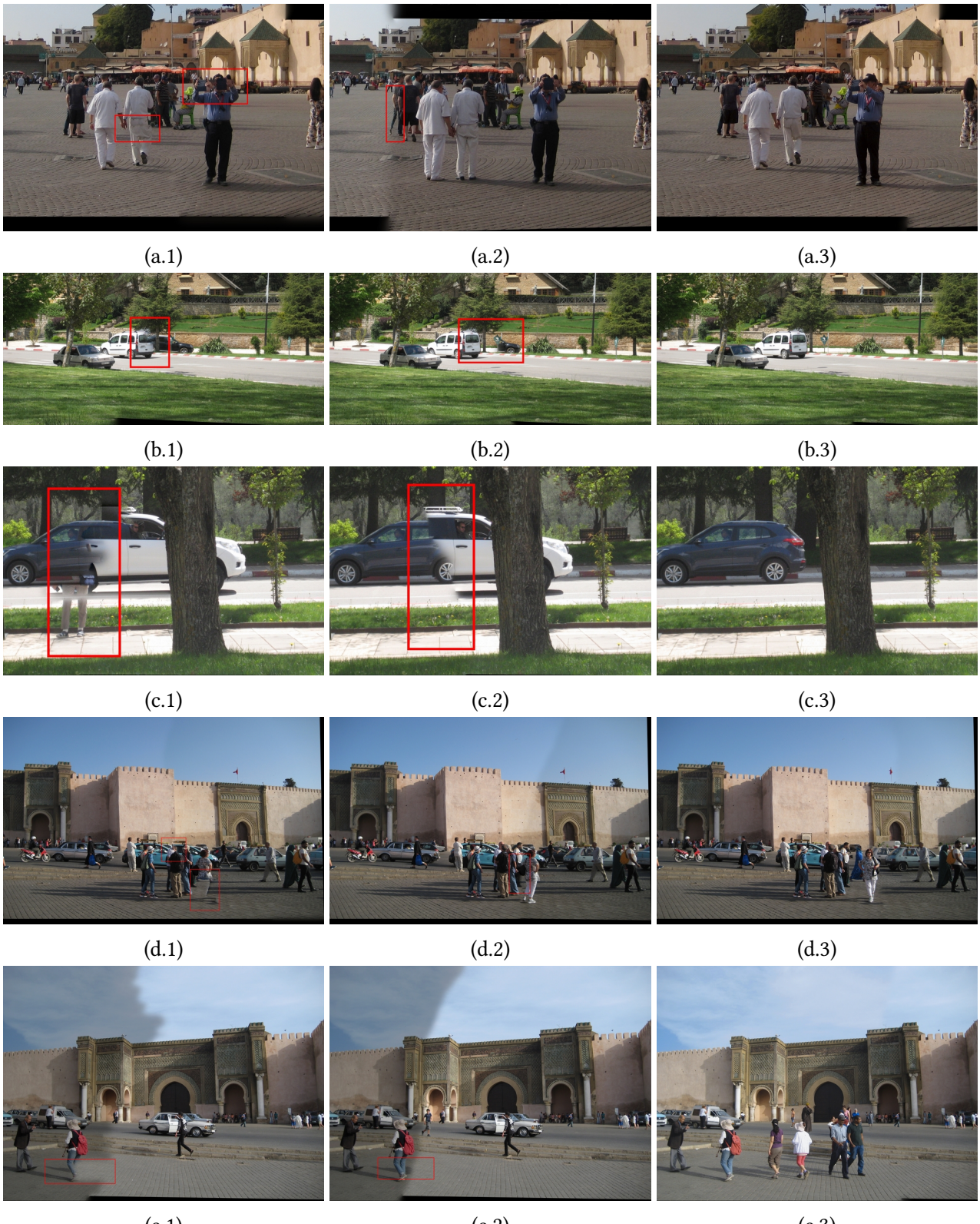


FIGURE 3.14 – "Seamlines" des images des données réelles de la figure 3.12 : (a) People (b) Valley (c) Cars (d) Plaza (e) Door, (1) Mills (2) CSDP (3) Notre Méthode.

La figure 3.15 affiche les résultats des mosaïques créées par les différentes approches. En plus, le tableau 3.4 présente les différentes mesures de comparaisons SSIM, SAM et IMR avec les résultats optimaux en gras. Les artefacts qui sont apparus dans la figure 3.15 sont encadrés en rouge.



(a.1) (a.2) (a.3)

(b.1) (b.2) (b.3)

(c.1) (c.2) (c.3)

(d.1) (d.2) (d.3)

(e.1) (e.2) (e.3)

FIGURE 3.15 – Mosaïques des images de la figure 3.12 (a) People (b) Valley (c) Cars (d) Plaza (e) Door, (1) Mills (2) CSDP (3) Notre Méthode.

	SSIM			SAM			IMR		
	Mills	CSDP	Notre méthode	Mills	CSDP	Notre méthode	Mills	CSDP	Notre méthode
(a)	0,9313	0,9787	**0,9962**	2,7483	4,2472	2,2114	0,9264	0,7981	0,9355
(b)	**0.9891**	**0.9925**	**0.9928**	**0.0598**	**0.1068**	**0,0002**	0.3879	0.6366	0.8319
(c)	**0.9851**	**0.9937**	**0.9874**	20.2969	18.3433	2.3855	0.8864	0.9444	0.9787
(d)	**0,9838**	**0,9828**	**0,9812**	2,7327	2,6623	**0,9624**	0,9514	0,9607	**0,9801**
(e)	**0,9836**	**0,9866**	**0,9846**	27,0865	29,9456	2,4402	0,9542	0,9586	**0,9909**

TABLE 3.4 – Valeurs SSIM, SAM et IMR pour les mosaïques créées à partir de la figure 3.12

A partir des résultats obtenus dans la figure 3.15 et le tableau 3.4, nous pouvons conclure que la méthode développée a créé des mosaïques de meilleur qualité avec une meilleur couleur et intensité. Ce résultat est confirmé avec les mesures de comparaison de la qualité de couleurs et d'intensité qui sont optimales même pour les images avec une différence d'exposure. En plus, la mesure de comparaison de l'intensité est toujours supérieure pour la méthode basée sur le détecteur de Canny par rapport aux autres méthodes.

Finalement, la comparaison des temps d'exécution des trois méthodes pour les données réelles est presenté dans la figure 3.16. La méthode développée est légèrement plus rapide que les autres méthodes. Nous notons la même remarque par rapport à la taille de l'image pour les données réelles que pour les données synthétiques où plus les images sont grandes plus la méthode développée est plus rapide par rapport aux autres méthodes. En outre, La taille des paires : 3.12.a qui a une taille de $(1300, 976)$, 3.12.b et 3.12.c avec une taille de $(1500, 1125)$, et 3.12.d et 3.12.e avec une taille de $(3072, 2304)$.

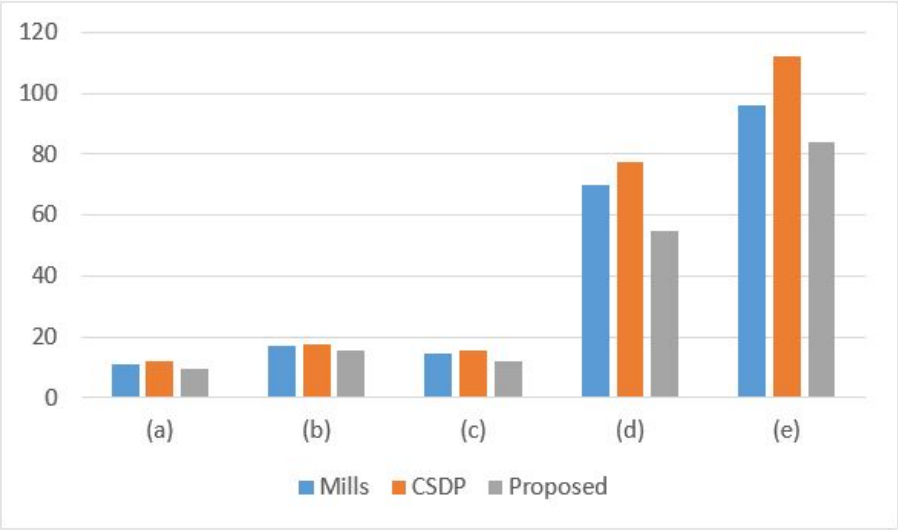


FIGURE 3.16 – Temps d'exécution des méthodes Mills, CSDP et la méthode développée pour les images de la figure 3.12.

3.3.3 Analyses des mesures obtenues

A partir des tableaux 3.2 et 3.4 , nous avons obtenu une amélioration de la qualité des résultats par rapport aux méthodes Mills et Dudek [54] et Zeng et al. [55] que ce soit sur des données synthétiques ou des données réelles selon les différentes mesures, nous remarquons en moyenne :

- Pour la mesure **SSIM** : une amélioration de +2% par rapport à la méthode Mills et Dudek et de +1% par rapport à la méthode CSDP de Zeng et al., ces valeurs ne sont pas élévées puisqu'ils ont été calculés sur les hautes fréquences.
- Pour la mesure **SAM** : une amélioration de +74,46% par rapport à la méthode Mills et Dudek et de +75,56% par rapport à la méthode CSDP de Zeng et al. : principalement grâce au transfert de couleurs utilisé comme prétraitement.
- Pour la mesure **IMR** : une amélioration de +22,84% par rapport à la méthode Mills et Dudek et de +16,22% par rapport à la méthode CSDP de Zeng et al.

Au niveau des temps d'exécutions, en moyenne la méthode développée a realisé un gain de temps de +16,00% par rapport à la méthode Mills et Dudek et de +22,35% par rapport à la méthode CSDP de Zeng et al.

3.4 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode de création de mosaïque d'images dynamiques basée sur un détecteur de contours pour le calcul d'une "seamline" optimale. Nous avons tout d'abord utilisé un prétraitement avec un transfert de couleurs pour régler la différence d'exposure entre les images et améliorer la qualité des images. Ensuite, nous avons généré une carte d'énergie en combinant entre une différence d'intensité et des régions créées en fusionnant les contours obtenus avec un détecteur de contours Canny et les "inliers" et "outliers" de la phase du recalage. Finalement, avec cette nouvelle carte d'énergie nous avons présenté un moyen de calculer une "seamline" d'une manière dynamique en la guidant vers les régions communes tout en évitant les zones qui peuvent créer des artefacts. L'efficacité et la robustesse de la méthode ont été testées avec différentes images dans différents cas, avec la présence d'exposition ou sans.

Chapitre 4

Mosaïque d'images basée sur le mouvement fractionnaire Brownien et l'algorithme A.*

4.1 Introduction . . . 72
4.2 Carte d'énergie fBm . . . 73
4.2.1 Recalage d'images . . . 75
4.2.2 Création de la carte d'énergie . . . 75
4.2.2.1 Mouvement fractionnaire Brownien . . . 75
4.2.2.2 "Zero Normalized Cross Correlation (ZNCC)" . . . 76
4.2.2.3 Génération de la carte d'énergie par FBm modifié et ZNCC . . . 77
4.3 Résultat mosaïque avec l'algorithme A* . . . 79
4.3.1 Calcul de la "seamline" A* . . . 79
4.3.2 "Pyramid blending" . . . 81
4.4 Analyses et résultats expérimentaux . . . 83
4.4.1 Analyses sur les paramètres . . . 83
4.4.1.1 Analyses sur le nombre de distributions . . . 84
4.4.1.2 Analyses sur la fréquence initiale . . . 86
4.4.1.3 Tests sur le chemin de la "seamline" . . . 87
4.4.2 Résultats expérimentaux . . . 88
4.4.3 Analyses des mesures obtenues . . . 94
4.5 Conclusion . . . 94

4.1 Introduction

Dans le chapitre précédent, nous avons présenté une méthode basée sur le détecteur de contours Canny. Cette dernière est très dépendante du recalage d'images vu qu'elle a besoin des "inliers" et "outliers" pour créer les régions. Ainsi, une grande zone de chevauchement est nécessaire pour l'obtention d'un bon résultat.

Dans ce chapitre, nous allons présenter une méthode basée sur l'algorithme A* pour calculer la "seamline" [78]. En effet, l'algorithme A* est un algorithme puissant qui trouve le chemin le plus court d'un point d'arrivée à un point de destination. Or, les cartes d'énergies qui existent dans la littérature ne permettent pas l'utilisation de cet algorithme. Ainsi, pour permettre l'utilisation de cette algorithme, nous avons généré une carte d'énergie basée sur le mouvement fractionnaire Brownien (fBm) calculé sur plusieurs distributions pour que la valeur de la carte d'énergie devienne unique. Les motivations pour la création de cette méthode sont :

- Premièrement, obtenir une bonne détection des objets dynamiques des images d'entrées en faisant en sorte que chaque valeur d'un pixel dans la carte d'énergie possède des informations sur son voisinage pour engendrer un calcul rapide de la "seamline" même dans les zones homogènes.
- Deuxièmement, détecter les erreurs qui apparaissent après la projection des images qui peuvent causer les effets parallaxes. Nous souhaitons utiliser plusieurs images pour tester au lieu d'utiliser que des paires d'images comme dans le chapitre précédent et ainsi nous voulons éviter l'accumulation des erreurs de projections.
- Troisièmement, introduire un algorithme d'apprentissage automatique basé sur les informations calculées dans la première étape pour trouver un chemin unique.
- Finalement, créer une mosaïque de haute qualité et facilement implémentable sans apparition d'artefacts.

Dans le but de réaliser ses motivations, les contributions effectuées sont :

- La première contribution consiste à résoudre le problème d'optimisation d'énergie en créant une carte d'énergie optimisée où chaque pixel reflète les valeurs de son voisinage en utilisant un mouvement fractionnaire Brownien ("fractional Brownian motion" (fBm)). En effet, le mouvement fractionnaire Brownien est la généralisation d'une fonction de distributions cumulatives qui utilise des pulsations aléatoires avec une distribution normale $N(0,1)$. Cependant, au lieu d'utiliser des pulsations aléatoires créées avec une fonction de bruit aléatoire, nous utiliserons la fonction de comparaison "Zero Normalized Cross Correlation (ZNCC)" qui retourne une valeur de similarité entre deux régions. Ce qui fait que dans chaque distribution la zone d'effet de la valeur de ZNCC est augmentée, puis avec plusieurs paramètres différents la cumulative des distributions est calculée pour obtenir la carte d'énergie finale. En outre, les raisons pourquoi

le fBm et ZNCC ont été utilisés ensemble sont :

(i) Chaque pixel aura une valeur unique dans la carte d'énergie car sa valeur est influencée par son voisinage.

(ii) Les zones homogènes n'auront pas les mêmes valeurs dans la carte d'énergie. Souvent les pixels avec les mêmes valeurs dans la carte d'énergie génèrent des artefacts plus facilement à cause de leur influence sur la "seamline".

(iii) L'obtention d'une meilleure détection des objets dynamiques et des erreurs qui apparaissent après la phase du recalage.

- La seconde contribution repose sur la recherche d'une "seamline" optimale en utilisant un algorithme d'apprentissage automatique. Cet algorithme est A* (A étoile). A* est un algorithme rapide qui est utilisé dans plusieurs domaines, cependant dans plusieurs méthodes de création de mosaïques, c'est impossible de l'utiliser à cause de la façon dont les cartes d'énergies ont été créées. S'il est utilisé, son temps d'exécution pour parcourir les cartes d'énergies est élevé, principalement à cause des zones homogènes. Ainsi, pour que nous puissions utiliser A*, la carte d'énergie a été transformée en un graphe où le point de départ et le point de destination sont choisis. Ce qui offre plusieurs avantages :

  (i) Premièrement, c'est possible de définir des nœuds comme des obstacles. Ainsi, en utilisant un seuil sur la carte d'énergie les objets dynamiques peuvent être considérés comme des obstacles impassables.

  (ii) Deuxièmement, A* utilise une fonction heuristique. Cette dernière calcule un coût qui estime la distance entre le nœud actuel et le nœud de destination. En la combinant avec les valeurs uniques calculées dans la carte d'énergie, le chemin devient plus simple à calculer.

  (iii) Finalement, puisque les pixels de la carte d'énergie contiennent des informations sur leur voisinage, ceci implique qu'il y aura un seul chemin unique calculé par A* n'importe le point de départ, ce qui va accélérer l'algorithme.

Le plan de ce chapitre est le suivant, nous allons tout d'abord expliquer comment calculer la carte d'énergie avec un mouvement fractionnaire Brownien modifié, puis nous allons calculer le chemin optimal de la "seamline" avec l'algorithme A*. Finalement, nous analyserons la méthode afin de choisir les meilleurs paramètres pour effectuer et tester la méthode sur différentes images en le comparant avec plusieurs méthodes. La figure 4.1 présente les différentes étapes de la méthode.

4.2 Carte d'énergie fBm

Dans cette section, nous allons tout d'abord présenter les algorithmes utilisés dans le recalage d'images, ensuite nous présenterons les différentes étapes pour la création de la carte d'énergie avec le mouvement fractionnaire Brownien en utilisant la fonction de similarité, Zero Normalized Cross Correlation. Puis, nous allons détecter les objets dynamiques.

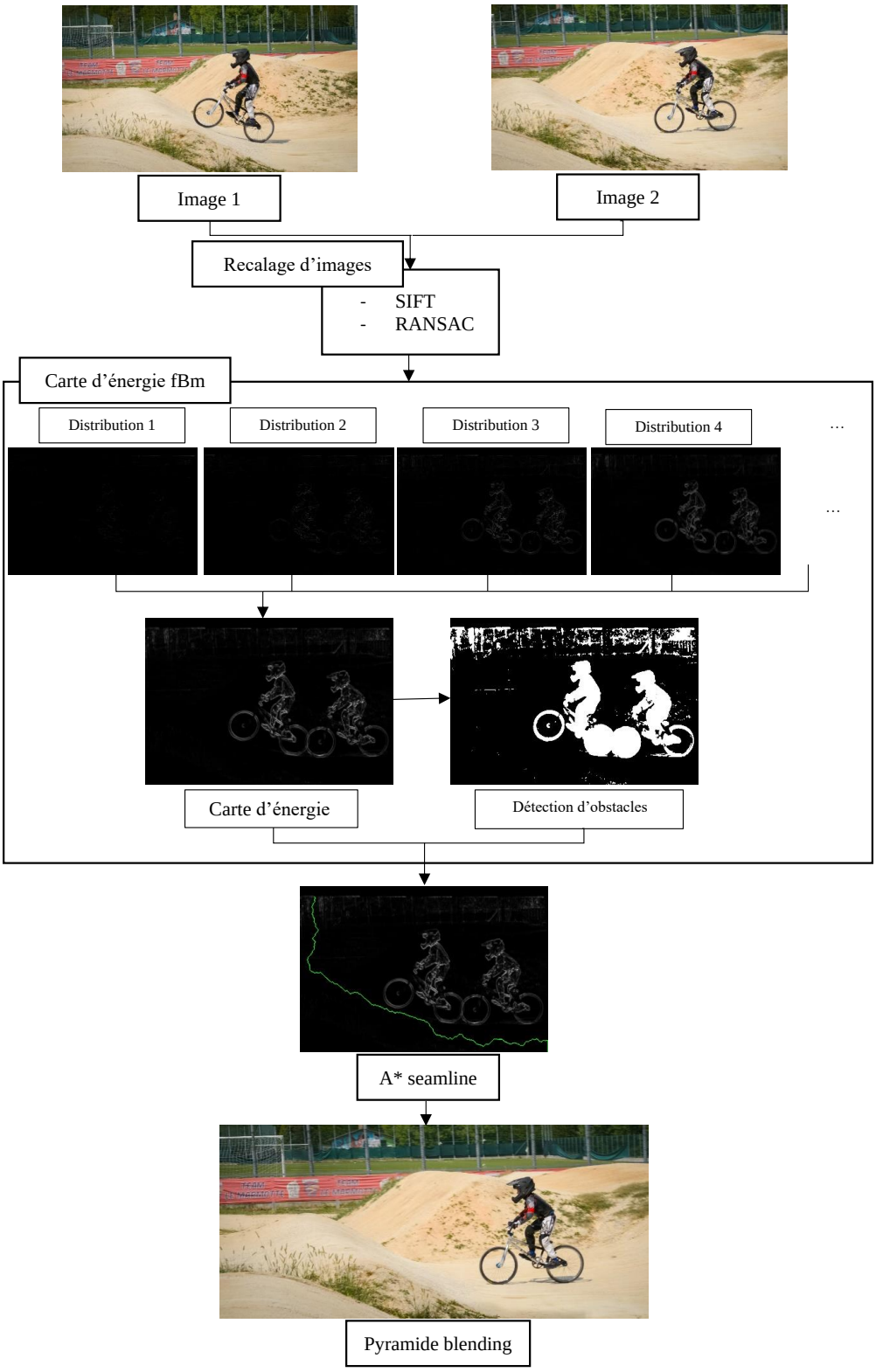


FIGURE 4.1 – Procédure pour la création d'une mosaïque avec la méthode basée sur le mouvement factionnaire Brownien et l'algorithme A*. [78].

4.2.1 Recalage d'images

De la même manière que le chapitre précédent, les algorithmes utilisés dans le recalage d'images sont SIFT [24] (Section 1.3.4.1) pour la détection et la descriptions des points d'intérêts et RANSAC [38] (Section 1.3.4.2) pour la mise en correspondance et le calcul de la transformation perspective.

C'est important de choisir un bon détecteur à cause des erreurs de projections qui apparaissent plus souvent quand la zone de chevauchement est petite où quand la taille des images est très grande. En plus, ces erreurs s'accumulent quand le nombre d'images utilisées est élevé.

4.2.2 Création de la carte d'énergie

L'objectif de cette sous-section est de créer la carte d'énergie pour une paire d'images en utilisant un mouvement fractionnaire Brownien (fBm) modifié.

4.2.2.1 Mouvement fractionnaire Brownien

Avant d'entamer le mouvement fractionnaire Brownien, nous allons tout d'abord définir le mouvement Brownien. En outre, un mouvement Brownien est une description mathématique du mouvement d'un point qui est dévié de son chemin suite à des fonctions de pulsations $W(t)$ aléatoires qui suivent une loi normale $N(0,1)$ à des intervalles irréguliers. Le Bm est aussi connu comme le processus de Wiener puisqu'il possède les mêmes propriétés, un processus stochastique à temps continu dont les accroissements sont stationnaires et indépendantes. De plus, ce processus est aussi Markovien (L'état actuel ne dépend pas de l'état précédent), martingale (La valeur estimée pour l'état future est la valeur de l'état actuel) et fractale (auto-similaire, irrégulier et fragmenté).

Ainsi, un mouvement fractionnaire Brownien (fBm) est défini comme une généralisation d'un mouvement Brownien (Bm). Plus précisemment, le fBm est une cumulative des distributions du mouvement Brownien avec un exposant de Hurst H. Cet exposant de Hurst est un paramètre $H \in [0,1]$ qui définit la dimension fractale du fBm où $Dim_{fBm} = Dim - H$.

Soit $(B(t))_{t\geq 0}$ un mouvement fractionnaire Brownien avec un exposant de Hurst $H \in [0,1]$. Mathématiquement, un fBm est défini comme :

- un processus continue.
- un processus Gaussien.
- Il possède des incréments stationnaires (tous les incréments ont la même loi). Ainsi, $\forall h > 0, (B(t+h) - B(t))_{t\geq 0}$ a la même loi que $B(t)_{t\geq 0}$.
- Il est auto-similaire : $\forall a > 0, (a^{-H}B(at))_{t\geq 0}$ a la même loi que $B(t)_{t\geq 0}$.

L'autosimilarité est la propriété la plus importante du fBm. Quand toutes les fonctions du fBm ont les mêmes fonctions de distributions finies, l'autosimilarité est réalisée.

Il existe plusieurs moyens [82] de simuler un fBm, ils sont divisés en deux catégories : les méthodes directes et les méthodes indirectes. Les méthodes directes sont basées sur une représentation stochastique [79], ou l'extraction d'une matrice de covariance [82] etc... Tandis que les méthodes indirectes [83] génèrent un bruit fractionnaire Gaussian, X, à des intervalles différentes $t = n/N$ avec $n = 1...N-1$ et additionnent les bruits pour créer un fBm. L'équation 4.1 formule la création d'un fBm avec une méthode indirecte.

$$B(t = n/N) = \sum_{k=0}^{n} X(t = k/N) \tag{4.1}$$

Où $n = 1...N-1$ avec $B(0) = 0$ et X est un bruit fractionnaire Gaussian généré à des intervalles de temps $t = n/N$.

4.2.2.2 "Zero Normalized Cross Correlation (ZNCC)"

Le but de la carte d'énergie qu'on souhaite créer est de surligner les différences dans les zones de chevauchements de deux images. Pour réaliser cet objectif, et ainsi détecter les objets dynamiques, nous allons utiliser le mouvement fractionnaire Brownien défini par l'équation 4.1. Cependant, au lieu d'utiliser une fonction de bruit aléatoire, nous calculerons le fBm avec une mesure de similarité. En effet, la mesure de similarité va être calculée autour du voisinage d'un pixel en spécifiant le rayon de la zone d'effet. Ainsi, pour chaque distribution, le rayon de la zone d'effet va augmenter à chaque octave et du coup chaque pixel aura une valeur plus détaillée sur son voisinage.

La mesure de similarité sélectionnée est la "Zero Normalized Cross Correlation" (ZNCC) [80]. Elle est souvent utilisée pour comparer deux images ou deux régions et retourne une valeur $\in [-1, 1]$. Quand deux images sont presque similaires, la valeur de ZNCC est proche de 1 et dans le cas contraire elle est proche de -1. L'équation 4.2 montre la formule de ZNCC.

$$ZNCC(Im_1, Im_2, x_1, y_1, x_2, y_2, r) = \frac{\sum_{i=-r}^{r} \sum_{j=-r}^{r} F(Im_1, x_1, y_1, i, j, r) \times F(Im_2, x_2, y_2, i, j, r)}{(2r+1)^2 \times \sigma_1(x_1, y_1, r) \times \sigma_2(x_2, y_2, r)} \tag{4.2}$$

Avec : $F(Im, x, y, i, j, r) = Im(x + i, y + j) - \mu(x, y, r)$

Où :

- Im_1 et Im_2 sont les images de chevauchements des images d'entrées en niveau de gris après la phase du recalage.
- x_1 et y_1, x_2 et y_2 sont les coordonnées des pixels qui vont être comparés des images Im_1 et Im_2 respectivement.
- r le rayon de la zone d'effet. Si la zone d'effet contient des pixels qui sont hors de la zone de l'image, ils ne sont pas pris en considération.

- $\mu_1(x_1, y_1, r)$ et $\mu_2(x_2, y_2, r)$ sont les moyennes des pixels de la zone d'effet des images Im_1 et Im_2 respectivement avec
  $\mu(x, y, r) = \frac{\sum_{i=-r}^{r} \sum_{j=-r}^{r} Im(x+i,y+j)}{(2r+1)^2}$.
- $\sigma_1(x_1, y_1, r)$ et $\sigma_2(x_2, y_2, r)$ sont les écarts types des pixels de la zone d'effet des images Im_1 et Im_2 respectivement avec
  $\sigma(x, y, r) = \sqrt{\frac{\sum_{i=-r}^{r} \sum_{j=-r}^{r} (Im(x+i,y+j)-\mu(x,y,r))^2}{(2r+1)^2}}$.

Puisque le but est de détecter la position des objets dynamiques, il faut changer l'intervalle de ZNCC pour qu'il devient un intervalle de $[0, 1]$ au lieu de $[-1, 1]$, et pour que les valeurs proches de 1 montrent les dissimilarités et les valeurs proches de 0 montrent les similarités (Ce qui est le contraire du ZNCC initial). L'équation 4.3 présente cette modification.

$$ZNCC_{fBm} = 1 - [1/2 * (ZNCC + 1)] \tag{4.3}$$

4.2.2.3 Génération de la carte d'énergie par FBm modifié et ZNCC

Pour pouvoir simuler un mouvement fractionnaire Brownien avec la Zero Normalized Cross Correlation, il faut définir des paramètres qui vont être utilisés dans chaque distribution pour obtenir l'effet fractionnaire et d'autosimilarité à la fonction. Ces paramètres sont la fréquence et la persistance.

Premièrement, le paramètre de la fréquence est défini par le rayon r de la zone d'effet du $ZNCC_{fBm}$. En effet, la valeur de $ZNCC_{fBm}$ est calculée pour tous les pixels avec un rayon prédéterminé. Ainsi, ce rayon va déterminer la fréquence initiale f, c'est à dire qu'à chaque fonction de distribution, la fréquence est mise à jour et le rayon augmente. Vu que $ZNCC_{fBm}$ ne peut pas être calculée pour des nombres réels, la fréquence est arrondie. Autrement, des interpolations doivent être effectuées.

Chaque fréquence est calculée avec $f_k = f_{k-1} * 2$. La première fréquence est le rayon initial $f_1 = r$. La figure 4.2 montre un exemple du processus avec un rayon initial de $r = 1$.

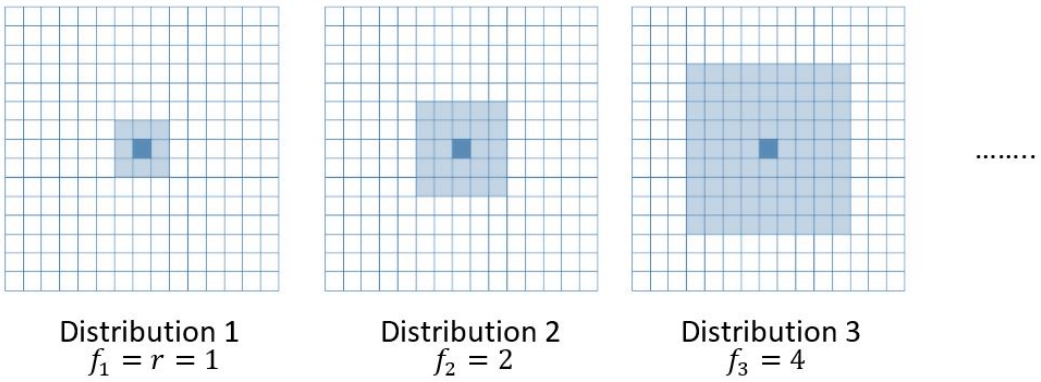


FIGURE 4.2 – Exemple de mise à jour de la fréquence à chaque distribution sur un pixel avec un rayon initial $r = 1$. Le pixel choisi est le pixel sombre au milieu.

Deuxièmement, un pixel aura des valeurs $ZNCC_{fBm}$ calculées à chaque distribution, par contre ces valeurs n'ont pas le même poids vu que la zone d'effet est différente entre les distributions ce qui peut négativement impacter la valeur finale (la cumulative des distributions) de la carte d'énergie. En effet, pour la première distribution, la valeur finale dépend des pixels qui sont proches, tandis que pour la dernière distribution, la valeur finale est impactée par des pixels qui sont loins. En outre, plus le nombre de distributions est élevé, plus la zone d'effet deviendra grande. Puisqu'ils existent différents objets dynamiques dans une scène, le paramètre de persistance p est utilisé pour contrôler l'effet des distributions.

Le paramètre de persistance p est introduit comme une valeur $p \in [0, 1]$ qui contrôle l'effet des distributions sur la valeur finale. Ainsi, une persistance proche de 0 veut dire que les premières distributions auront un plus grand impact sur la valeur finale tandis qu'une valeur proche de 1 veut dire que ce sera les dernières distributions qui impacteront plus la valeur finale. Cette valeur permet d'ajouter un contrôle à la méthode. Effectivement, si les images possèdent peu d'objets dynamiques une valeur de persistance élevée est préférable, cependant pour les images avec beaucoup d'objets dynamiques une valeur proche de 0 est meilleure. A noter que c'est possible d'ignorer ce paramètre seulement si un nombre de distributions adapté pour les images est choisi.

Ainsi la carte d'énergie finale est calculée par l'équation 4.4.

$$E(x, y) = \sum_{k=1}^{D} p_k * ZNCC_{fBm}(Im_1, Im_2, x, y, x, y, f_k) \tag{4.4}$$

Où :

- D est le nombre de distributions.
- p_k le paramètre de persistance avec $p_k = p_{initial}^k$ et $p_{initial}$ la persistance initiale choisie.
- Im_1 et Im_2 la zone de transition au niveau de gris des images d'entrées après la projection.
- x et y sont les coordonnées des images Im_1 et Im_2. Les images ont la même taille du coup $ZNCC_{fBm}$ est calculée pour les mêmes coordonnées.
- f_k est la fréquence et le rayon pour $ZNCC_{fBm}$ avec $f_k = f_{k-1} * 2$ et $f_1 = r$ le rayon initial.

Soit I_1 et I_2 deux images utilisées pour créer une carte d'énergie fBm. La figure 4.3 présente le processus de création avec $D = 3$ distributions avec une persistance initiale $p_{initial} = 0.75$ et un rayon initial $r = 1$.



(a)



(b)



(c)

FIGURE 4.3 – Création de la carte d'énergie (a) I_1 et I_2 les images d'entrées, (b) Trois distributions calculées avec $ZNCC_{fBm}$ en utilisant la zone de chevauchement des images I_1 et I_2. De gauche à droite, première distribution avec un rayon initial de $f_1 = r = 1$, la seconde $f_2 = 2$, et la dernière distribution $f_3 = 4$ (c) Carte d'énergie E avec une persistance de $p = 0.75$.

4.3 Résultat mosaïque avec l'algorithme A*

Dans le but de générer la mosaïque à partir de la carte d'énergie calculée dans l'étape précédente, une "seamline" optimale est calculée sur cette carte avec l'algorithme A* [81] en considérant les objets dynamiques comme des obstacles impassables. Ainsi, dans cette section, nous allons présenter la méthode de calcul de la "seamline" et le "blending" d'images utilisé.

4.3.1 Calcul de la "seamline" A*

Afin que la mosaïque générée ne contienne aucun artefacts, il est nécessaire de calculer une "seamline" qui ne crée aucune discontinuité. Néanmoins, la majorité des approches utilisent soit une méthode qui calcule la "seamline" d'une manière linéaire ou ont une mauvaise détection des objets dy-

namiques. Par conséquence, nous avons opté pour l'utilisation de l'algorithme A* [81], une méthode d'apprentissage automatique qui va combiner entre la carte d'énergie précédemment calculée et une heuristique pour estimer un coût de mouvement qui va guider cette "seamline".

Pour utiliser A*, nous considérerons la carte d'énergie E comme un graphe où chaque nœud est connecté aux 8 nœuds de son voisinage et contient sa valeur correspondante.

A* est un algorithme de recherche de chemin qui va traverser le graphe d'un point de départ à un point d'arrivée. Premièrement, l'algorithme calcule le coût de passage entre chaque nœud adjacent en additionnant la valeur du nœud obtenue dans la carte d'énergie et l'heuristique. Puis, A* choisi le nœud le plus bénéfique à traverser (Le nœud avec le coût le plus bas) jusqu'à arriver au nœud de destination. L'équation 4.5 montre la formule pour calculer le coût.

$$cost(x, y) = 100 * (E(x, y) + h(x, y, x_{dest}, y_{dest})) \tag{4.5}$$

Où :

- x et y sont les coordonnées d'un nœud.
- x_{dest} et y_{dest} sont les coordonnées du nœud de destination.
- E la carte d'énergie.
- h une fonction heuristique. Les distances euclidienne, Manhattan ou diagonale peuvent être utilisées comme heuristique avec la distance diagonale étant la plus rapide.

Avant d'effectuer la recherche de la meilleure "seamline", une carte d'obstacle est générée. En effet, l'un des avantages de l'algorithme A* c'est qu'il ne passera jamais par des nœuds qui sont considérés comme obstacles. En outre, pour décider si un nœud est un obstacle ou pas, un seuil $Thrd \in [0.1, 0.3]$ est utilisé sur la carte d'énergie E pour créer la carte d'obstacle. Le choix d'un seuil bas est à cause du faite que $ZNCC_{fBm}$ est une méthode précise dans la comparaison des zones similaires et retourne des valeurs basses. Finalement, après la création de la carte d'obstacle, par mesure de précaution une dilatation est effectuée pour fermer les trous qui peuvent exister pour éviter un faux départ où la "seamline" restera coincée dans une zone fermée par un obstacle. La figure 4.4.a montre la carte d'obstacle obtenue avec la carte d'énergie de la figure 4.3.c.

Dans le but de trouver un chemin optimal, un point de départ et un point de destination doivent être définis. Ces points sont choisis parmi les points de la carte d'énergie E. En effet, le point de départ peut être le point ayant la valeur la plus basse de la première ligne de E où n'importe quel autre point de la première ligne qui ne fait pas parti de la carte d'obstacle. Le choix de ce point n'impacte pas la recherche du chemin puisque l'algorithme ne peux pas prédire qu'elle est le meilleur point de départ. En plus, la recherche du point de départ est triviale vu qu'à un certain moment dans la recherche du meilleur chemin, les chemins de points de départs différents deviendront unique et fusionneront en un seul chemin n'importe le point de départ. La figure 4.4 présente un exemple où les "seamlines" après une certaine étape suivent le même chemin, avec la "seamline" de la figure 4.4.b qui possède

comme point de départ le point ayant la valeur la plus petite et la "seamline" de la figure 4.4.c qui possède comme point de départ le premier point qui n'est pas un obstacle. Cependant, pour le point de destination, le point avec la valeur la plus petite de la dernière ligne est choisi. Ce point est unique dans la carte d'énergie et le résultat final est gravement influencé par le point de destination contrairement au point de départ.



(a) (b) (c)

FIGURE 4.4 – Calcul de la "seamline" avec A* (a) Carte d'obstacle extraite de la carte d'énergie E de la figure 4.3.c, (b) "Seamline" A* qui commence par le point qui a la valeur la plus petite de la première ligne (c) "Seamline" A* qui commence par le premier point qui n'est pas un obstacle.

Pour atteindre le nœud de destination à partir du nœud de départ et trouver le meilleur chemin avec l'algorithme A*, deux listes sont utilisées : la première qui stocke tous les chemins possibles qui commencent par le nœud de départ et la deuxième qui marque si un nœud a déjà été visité ou non. Les étapes pour calculer la meilleure "seamline" sont les suivantes : (Nous supposons qu'un chemin existe)

1) Le nœud actuel est marqué comme visité (dans le cas de la première itération ce nœud est le nœud de départ) et dans son voisinage une vérification est effectuée sur les nœuds pour voir si l'un d'entre eux n'est pas le nœud de destination. Si cette condition est vraie alors l'algorithme se termine.
2) La valeur du coût (Equation 4.5) est calculée pour tous les nœuds non visités du voisinage du nœud actuel qui ne font pas partie de la carte d'obstacle et chaque chemin est stocké dans la liste des chemins. En plus, tous les nœuds dont le coût a été calculé sont marqués comme visités.
3) Dans la liste des chemins, la somme des coûts des nœuds de chaque chemin est calculée et le chemin qui a la somme la plus petite, son dernier nœud est choisi comme le nœud actuel.
4) Les étapes 1,2 et 3 se répètent jusqu'à trouver le nœud de destination.

Le chemin final obtenu par l'algorithme A* est choisi comme la "seamline" optimale de la carte d'énergie.

4.3.2 "Pyramid blending"

Le "blending" utilisé avec la "seamline" A* est le "pyramid blending" (Section 1.4.1.3). L'image masque de la méthode est créée en affectant une valeur de 255 aux pixels à droite de la "seamline".

La figure 4.5 présente la mosaïque obtenue avec les images d'entrées de la figure 4.3.



(a) (b)

FIGURE 4.5 – Résultat mosaïque après le "pyramid blending" (a) avec bordures (b) sans bordures.

La figure 4.6 résume la procédure de création de la mosaïque basée sur un fBm modifié avec ZNCC.

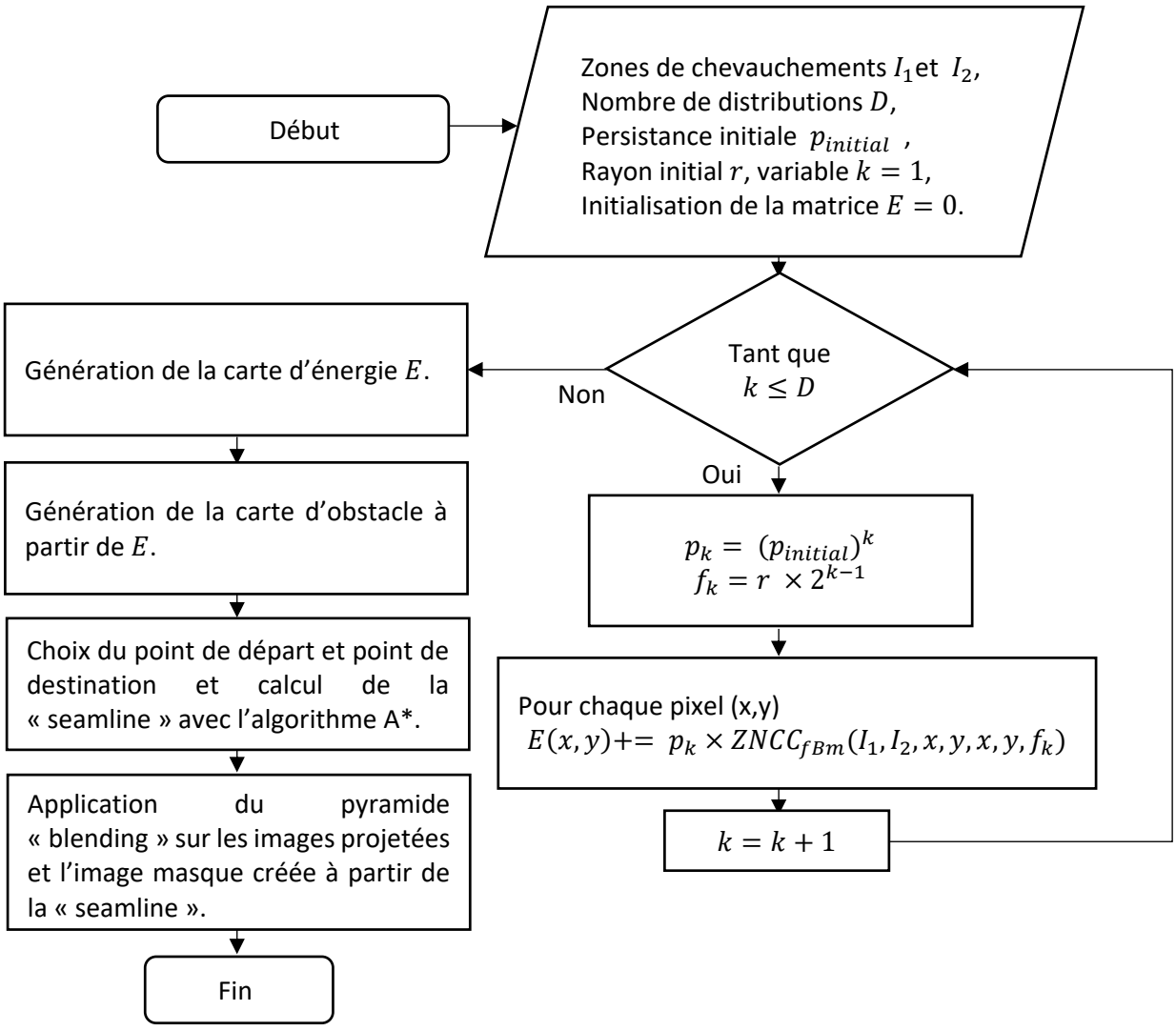


FIGURE 4.6 – Algorithme de la méthode développée basée sur l'utilisation d'un fBm modifié avec ZNCC

4.4 Analyses et résultats expérimentaux

Dans cette section, différentes images ont été utilisées pour analyser et tester la méthode développée basée sur un fBm modifié et l'algorithme A*. Tout d'abord, dans la première sous-section, puisque différents paramètres sont utilisés pour créer la carte d'énergie, ces paramètres seront analysés pour montrer l'impact qu'ils ont sur la détection des objets dynamiques, le calcul de la "seamline" et du résultat mosaïque. Ensuite, dans la deuxième sous-section, les résultats expérimentaux seront présentés et la méthode développée est comparée avec plusieurs méthodes.

La majorité des images utilisées dans ce chapitre appartiennent soit à la base de données DAVIS [77] (Figure 4.1, 4.3.a) ou sont capturées avec une caméra digitale Canon Digital IXUS 60. Les différentes méthodes ont été codées en Java avec la librairie OpenCV sur un PC portable ayant un microprocesseur Intel Core i7-3632QM CPU à 2.20 GHz.

4.4.1 Analyses sur les paramètres

Après la projection des images dans le même système de coordonnées, la prochaine étape est la création de la carte d'énergie en utilisant un mouvement fractionnaire Brownien avec une similarité "Zero Normalized Cross Correlation". Cependant, des paramètres comme le nombre de distributions et la fréquence (rayon de la zone d'effet ZNCC) impactent la carte d'énergie. Effectivement, d'une part, choisir un nombre trop élevé peut augmenter la taille des zones d'objets dynamiques, ce qui bloquera la "seamline" avec la carte d'obstacle et augmentera inutilement le temps d'exécution de la carte d'énergie. De l'autre part, choisir des nombres trop bas va réduire la précision de la détection des objets dynamiques, ce qui augmentera le temps d'exécution de la recherche de la "seamline". Par conséquence, choisir un nombre adapté pour le nombre de distributions et la fréquence va permettre d'obtenir des résultats optimaux au niveau graphique et en temps d'exécution.

Dans cette méthode, la valeur de persistance utilisée est $p = 0.75$. Cette valeur a été choisie pour permettre aux distributions élevées d'impacter la carte d'énergie sans gêner le calcul du chemin de la "seamline". De plus, comme précisé précédemment, la valeur de persistance peut être ignorée si le nombre de distributions choisi est optimal.

La figure 4.7 contient deux séries d'images qui vont être utilisées pour différentes analyses, la paire 4.7.a, qui fait partie de la base de donnée DAVIS [77], contient un mouvement horizontal, tandis que la paire d'images 4.7.b, qui a été capturée, contient différents mouvements orthogonaux.

Les deux séries de la figure 4.7 sont utilisées pour :

- Premièrement, montrer les effets du nombre de distributions sur la carte d'énergie, le chemin de la "seamline" et l'impact sur le temps d'exécution avec une fréquence initiale de 1.
- Deuxièmement, montrer les différentes cartes d'énergie en variant la fréquence. En effet, une fréquence de 2 peut être obtenue soit avec un nombre de distributions de 2 avec une fréquence initiale de 1, soit avec un nombre de distributions de 1 avec une fréquence initiale de 2.

- Finalement, montrer le résultat mosaïque de la méthode en ajoutant différents bruits ou expositions ou une combinaison des deux et leurs effets sur le chemin de la "seamline" et le temps d'exécution.



(a) (b)

FIGURE 4.7 – Images utilisées pour l'analyse (a) Première série de la base DAVIS (bmx-trees) (b) Deuxième série Plaza.

4.4.1.1 Analyses sur le nombre de distributions

La première analyse sur les images de la figure 4.7 a été effectuée avec 5 distributions maximum avec une fréquence initiale de 1 et une persistance de 0.75.

Dans la figure 4.8, on a testé l'impact de différentes distributions sur les images de la figure 4.7.

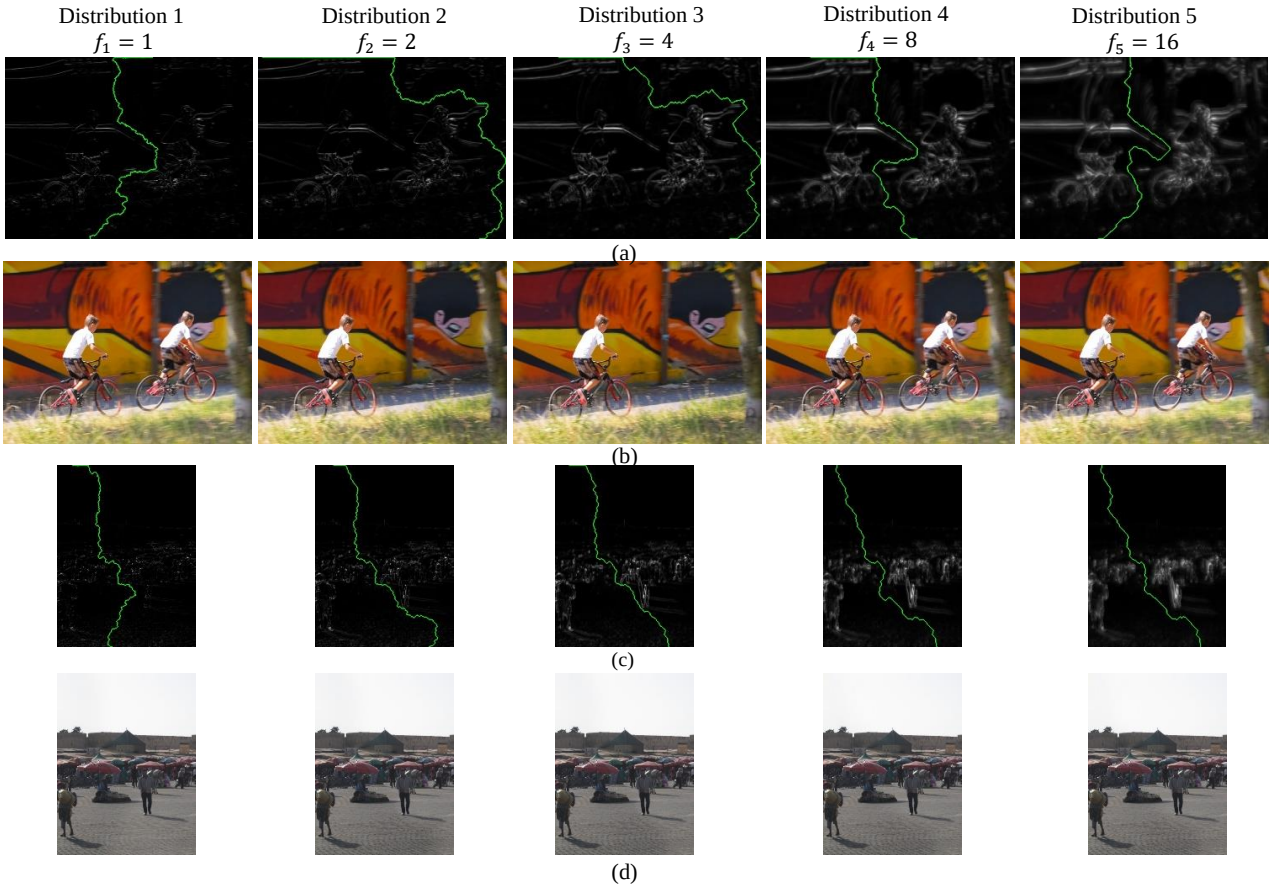
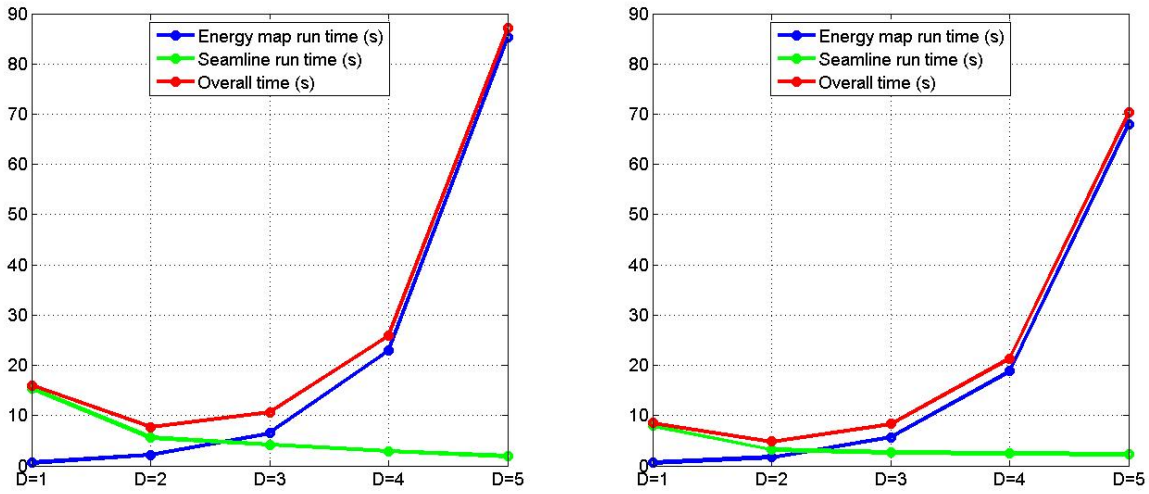


FIGURE 4.8 – Cartes d'énergie (a, c) et zones de chevauchements des résultats (b, d) dans différents nombre de distributions créés à partir des images de la figure 4.7.

Chaque colonne de cette figure montre la carte d'énergie avec la "seamline" et la zone de chevauchement de la mosaïque résultante dans différentes nombre de distributions allant de 1 jusqu'à 5. Plusieurs remarques peuvent être notées :

- Pour la première distribution où $D = 1$, la carte d'énergie calculée affiche les objets dynamiques, cependant les valeurs ne sont pas assez précises, par conséquence le calcul de la "seamline" possèdera un temps d'exécution élevé. La faible précision a causé une duplication d'objets dans le résultat.
- Avec l'augmentation du nombre de distributions à 2 ou 3, la carte d'énergie devient plus détaillée ce qui a rendu la "seamline" plus précise et du coup la duplication d'objets n'existe plus dans le résultat.
- Quand le nombre de distributions est passé à 4 ou 5, les objets dynamiques ont impacté la carte d'énergie plus fortement, ce qui a falsifié le chemin que la "seamline" a pu prendre en enlevant des chemins sûrs. En effet, quand le nombre de distributions devient trop élevé, ça peut causer la création d'une ligne verticale dans la carte d'obstacle qui causera l'échec du calcul de la "seamline" puisque cette dernière ne passera pas par des obstacles. Le seul moyen pour régler cette erreur est de réduire la valeur de la persistance pour que les distributions élevées n'impacteront plus la carte d'énergie fortement lors du calcul de la cumulative.



(a) (b)

FIGURE 4.9 – Temps d'exécution pour la création de la carte d'énergie et le calcul de la "seamline", ainsi que leur somme pour (a) Figure 4.7.a et (b) Figure 4.7.b sur différentes distributions.

Pour comparer les résultats obtenus dans les différentes distributions, la figure 4.9 montre le temps d'exécution de la carte d'énergie et du calcul de la "seamline". En analysant les résultats, nous remarquons que plus le nombre de distributions utilisées est élevé plus le temps d'exécution de la création de la carte d'énergie augmente, cependant pour le temps d'exécution du calcul de la "seamline" il décroît de plus en plus. Cette observation est expliquée par la carte d'énergie qui devient de plus en

plus détaillée. Le point d'intersection des deux courbes peut être observé entre un nombre de distributions entre 2 et 3. A ce point d'intersection l'algorithme devient rapide avec une somme des temps d'exécutions de $7,723s$ pour la figure 4.7.a et $4,875s$ pour la figure 4.7.b.

En combinant les résultats de la figure 4.8 et la figure 4.9, nous pouvons présumer que les meilleurs résultats visuels et en temps d'exécutions sont atteignables avec une distribution de 2 ou 3. Cependant c'est possible de réduire le nombre de distributions en réduisant la fréquence initiale, ainsi il faut analyser l'influence de la fréquence initiale par rapport au nombre de distributions.

4.4.1.2 Analyses sur la fréquence initiale

En réduisant le nombre de distributions et en augmentant le nombre de la fréquence initiale, c'est possible d'obtenir la même fréquence maximale, cependant la même précision dans la carte d'énergie n'est pas obtenable. Vu que nous avons établi que le nombre de distributions optimal est soit 2 ou 3, la carte d'énergie et la "seamline" de la figure 4.7.a sont comparées que pour ces distributions. La figure 4.10 montre les différentes cartes d'énergie en utilisant différentes fréquences et distributions. A partir de cette figure, nous pouvons remarquer que les meilleurs "seamlines" obtenues sont les "seamlines" qui ont été calculées avec une carte d'énergie possédant une fréquence initiale de 1. Ce sont les seules "seamlines" qui n'ont pas créé de duplication d'objets. En effet, quand une carte d'énergie est calculée avec une fréquence initiale de 1, la première distribution contient le plus d'informations sur les pixels des images sans qu'ils soient influencés par leur voisinage.

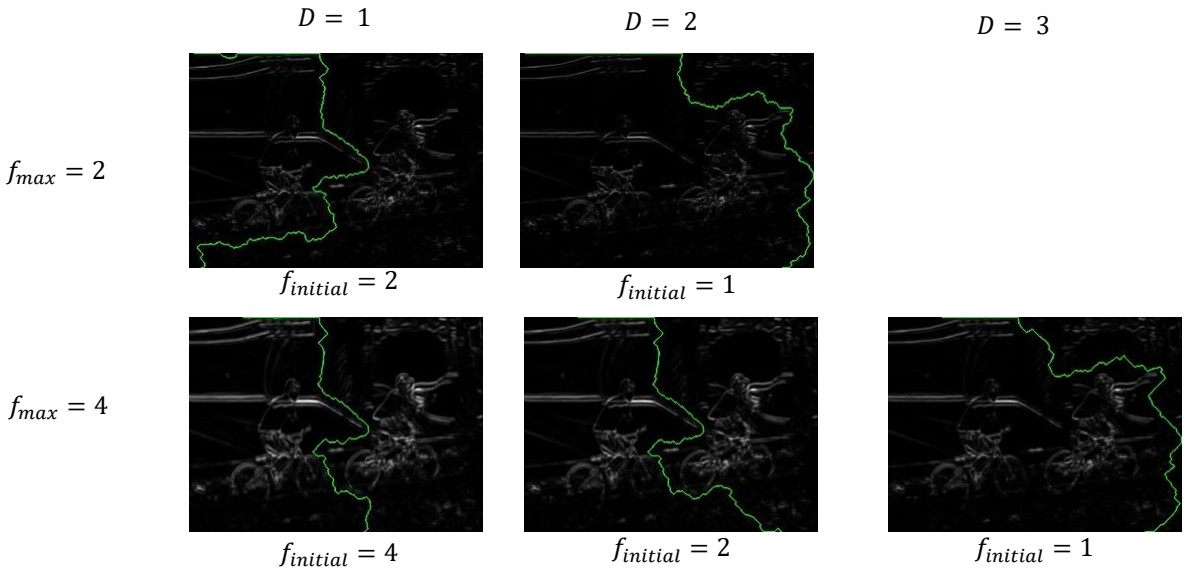


FIGURE 4.10 – Cartes d'énergie et les "seamlines" de la série d'images Figure 4.7.a sous différentes fréquences et distributions.

La figure 4.11 présente les différents temps d'exécutions pour la carte d'énergie et le calcul de la "seamline" à différentes fréquences maximales. Dans cette figure, nous avons testé le temps d'exécution sur trois fréquences maximales : 2, 4 et 8. Dans chaque fréquence maximale, c'est possible

d'obtenir une carte d'énergie avec différentes combinaisons entre le nombre de distributions et la fréquence initiale. Par exemple, pour une fréquence maximale de 4, plusieurs combinaisons de cartes d'énergie existent, soit une fréquence initiale de $f_{initial} = 1$ avec un nombre de distributions $D = 3$, ou $f_{initial} = 2$ avec $D = 2$ ou $f_{initial} = 4$ avec $D = 1$.

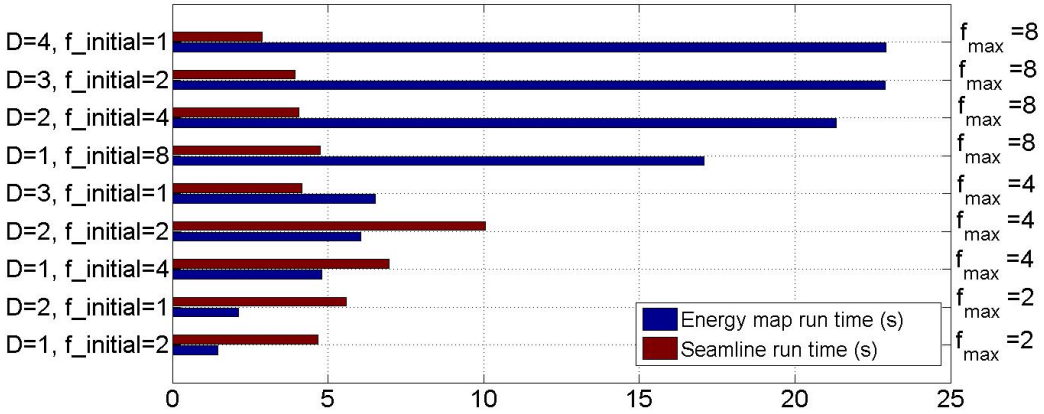


FIGURE 4.11 – Temps d'éxecution de la création de la carte d'énergie et du calcul de la "seamline" en secondes pour la série de la figure 4.7.a avec différentes fréquences maximales en variant la fréquence initiale et le nombre de distributions.

Les résultats de la figure 4.11 montrent que à chaque fréquence maximale ($f_{max} = 2$, 4 ou 8), le temps d'exécution est plus rapide quand le nombre de distributions est $D = 1$. Cependant, dans ce cas-là, vu que le nombre de distributions est 1, le paramètre de persistance n'as plus d'effet ce qui résultera aux mêmes problèmes quand le nombre de distributions est élevé où chaque pixel de la carte d'énergie est influencé fortement par son voisinage.

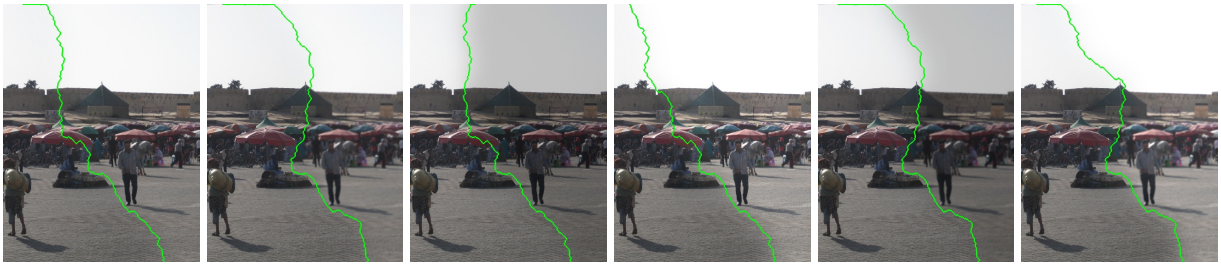
En combinant les résultats de la figure 4.10 et la figure 4.11, nous présumons que la carte d'énergie est le plus précise avec un nombre de distributions qui a une fréquence initiale de 1 malgré le coût en temps d'exécution. Cette observation est confirmée par la figure 4.10 où les "seamlines" qui n'ont pas créés de duplication d'objets sont ceux avec une fréquence initiale de 1.

4.4.1.3 Tests sur le chemin de la "seamline"

Plusieurs algorithmes ont des problèmes dans le calcul de la "seamline" quand un bruit, exposure, une différence d'intensité, etc... existent dans les images d'entrées. Du coup, une étape de prétraitement est souvent ajoutée pour ajuster ces différences pour que la "seamline" ne donne pas un mauvais résultat et pour ne pas trop perdre en temps d'exécution. Afin de tester la robustesse de la méthode developpée et pour vérifier qu'aucun n'artefact n'est créé, la seconde image de la série de la figure 4.7 est modifiée en ajoutant un bruit, une sous-exposition, une sur-exposition ou une combinaison de ces modifications.

Les images affichées dans la figure 4.12 montrent que le résultat obtenu n'as pas changé. En effet, la méthode développée n'a créé aucun artefacts à cause la façon dont la carte d'énergie est calculée

avec ZNCC. En plus, nous pouvons remarquer que les "seamlines" sont presque toutes similaires avec presque le même temps d'exécution.



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

FIGURE 4.12 – Zones de chevauchements des mosaïques résultantes de la figure 4.7.b où : (a) Aucune modification, (b) Bruit, (c) Sous-exposition, (d) Sur-exposition, (e) Bruit + Sous-exposition, (f) Buit + sur-exposition. Avec un temps d'exécution pour le calcul de la "seamline" de (a) $2,409s$ (b) $2,438s$ (c) $2,501s$ (d) $2,156s$ (e) $2,393s$ (f) $2,085s$.

4.4.2 Résultats expérimentaux

Pour montrer l'efficacité de la méthode développée, nous allons utiliser cinq séquences d'images où chaque séquence est utilisée pour faire un test particulier, puis nous allons comparer les résultats obtenus avec les résultats des logiciels **Hugin** et **PTGui**, ainsi que les méthodes : la méthode **CSDP** de Zeng et al. [55] (Section 2.3.1.2), la méthode **Graphcut** de Li et al. [56] (Section 2.3.1.4) et la méthode **Inlier/Outlier** présenté dans le chapitre 3 [71] qui est basée sur la création de régions avec le détecteur Canny.

Les méthodes sont comparées avec les mesures de qualité "Structure Similarity Index Measure" (SSIM) [66] (Section 2.4.2.1), le "Multi-Scale Structure Similarity Index Measure" (MS-SSIM) [67] (Section 2.4.2.2) et la valeur de SSIM que sur les points de la "seamline" (SSIM-Seamline). Finalement, le temps d'exécution de la création de la mosaïque a été calculé pour chaque méthode.

Les logiciels utilisés pour la comparaison avec la méthode développée (Hugin et PTGui) sont tous les deux des logiciels à jour avec PTGui qui est un logiciel professionnel.

La majorité des chercheurs quand ils veulent tester leurs algorithmes, ils utilisent des images qui contiennent soit un seul objet dynamique qui est visible sur la majorité de l'avant plan avec quelques objets dynamiques presque invisible dans la scène. Ainsi, leurs méthodes génèrent des erreurs quand des occlusions ou trop d'objets dynamiques existent dans la zone de chevauchement.

Différentes séquences vont être utilisées pour montrer la performance des algorithmes sur des aspects différents. Les séquences présentées dans la figure 4.13 sont :

(a),(b) Deux séquences de 3 images avec un seul objet dynamique avec la présence d'une occlusion, la première séquence (a) contient des textures dynamiques (feuilles, herbes) et du flou tandis

que la deuxième (b) contient dans la majorité des textures statiques avec des reflets de l'objet dynamique présent dans les images.

(c) Une série de 3 images où certains objets dynamiques sont coupés sans la présence de la partie manquante.

(d) Une séquence de 3 images d'une scène à l'intérieur avec beaucoup d'objets dynamiques.

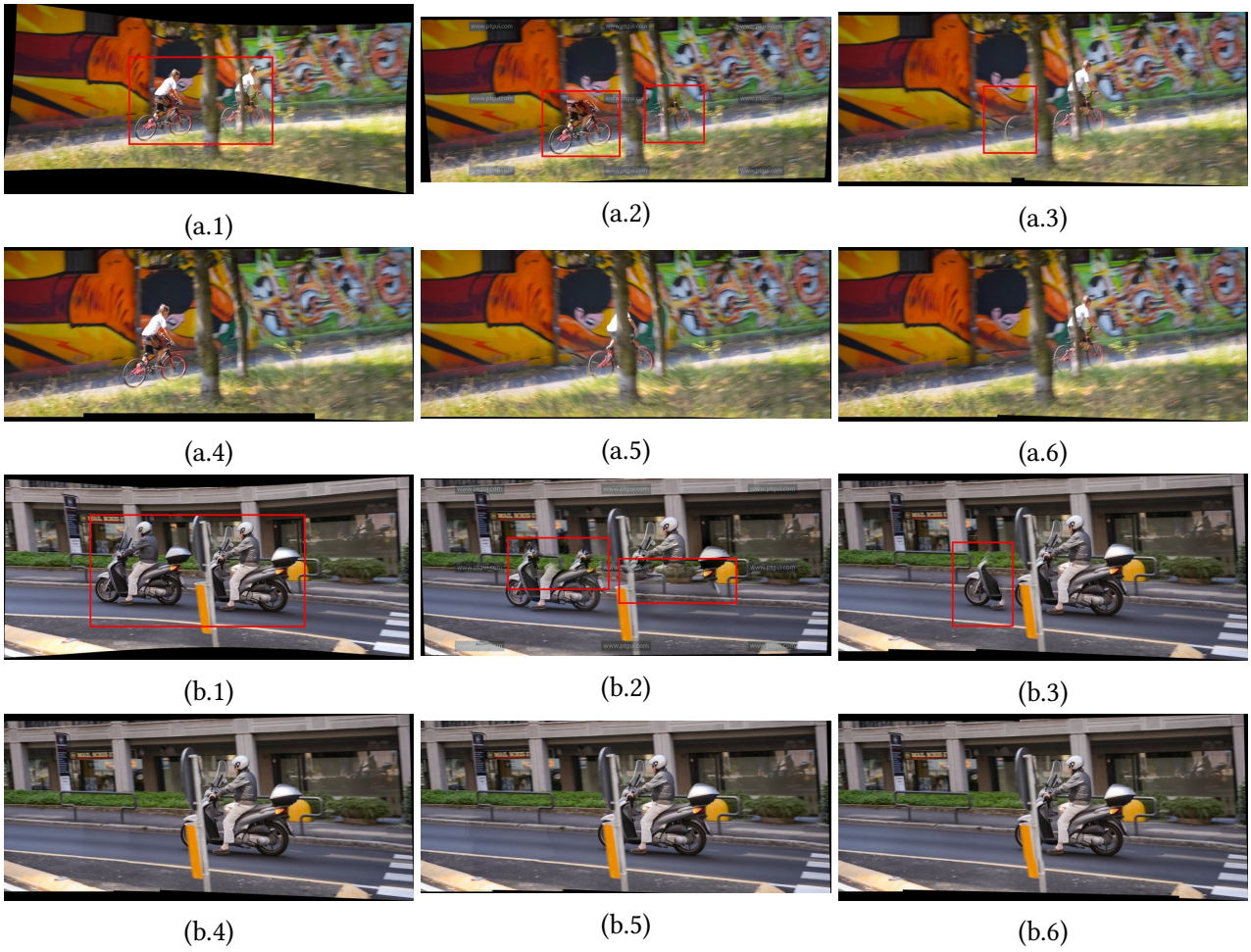
(e) Une série d'images avec un nombre élevé d'objets dynamiques et de textures (nuages).

Les séries d'images de la figure 4.13.a, 4.13.b et 4.13.d font partie de la base de données DAVIS [77] tandis que les figures 4.13.c et 4.13.e ont été capturées.



FIGURE 4.13 – Séries utilisées pour la partie expérimentale, (a) Bmx-trees, (b) Scooter-gray, (c) Valley, (d) Lindy-hop and (e) Bridge

Les images de la figure 4.14 montrent les résultats mosaïques des séquences 4.13.a et 4.13.b. Le premier test a été effectué sur deux séquences de 3 images avec un seul objet dynamique pour tester les occlusions sur 4.13.a qui possède des textures dynamiques (feuilles, herbes) tandis que la deuxième séquence 4.13.b a possède des textures statiques.



(a.1) (a.2) (a.3)

(a.4) (a.5) (a.6)

(b.1) (b.2) (b.3)

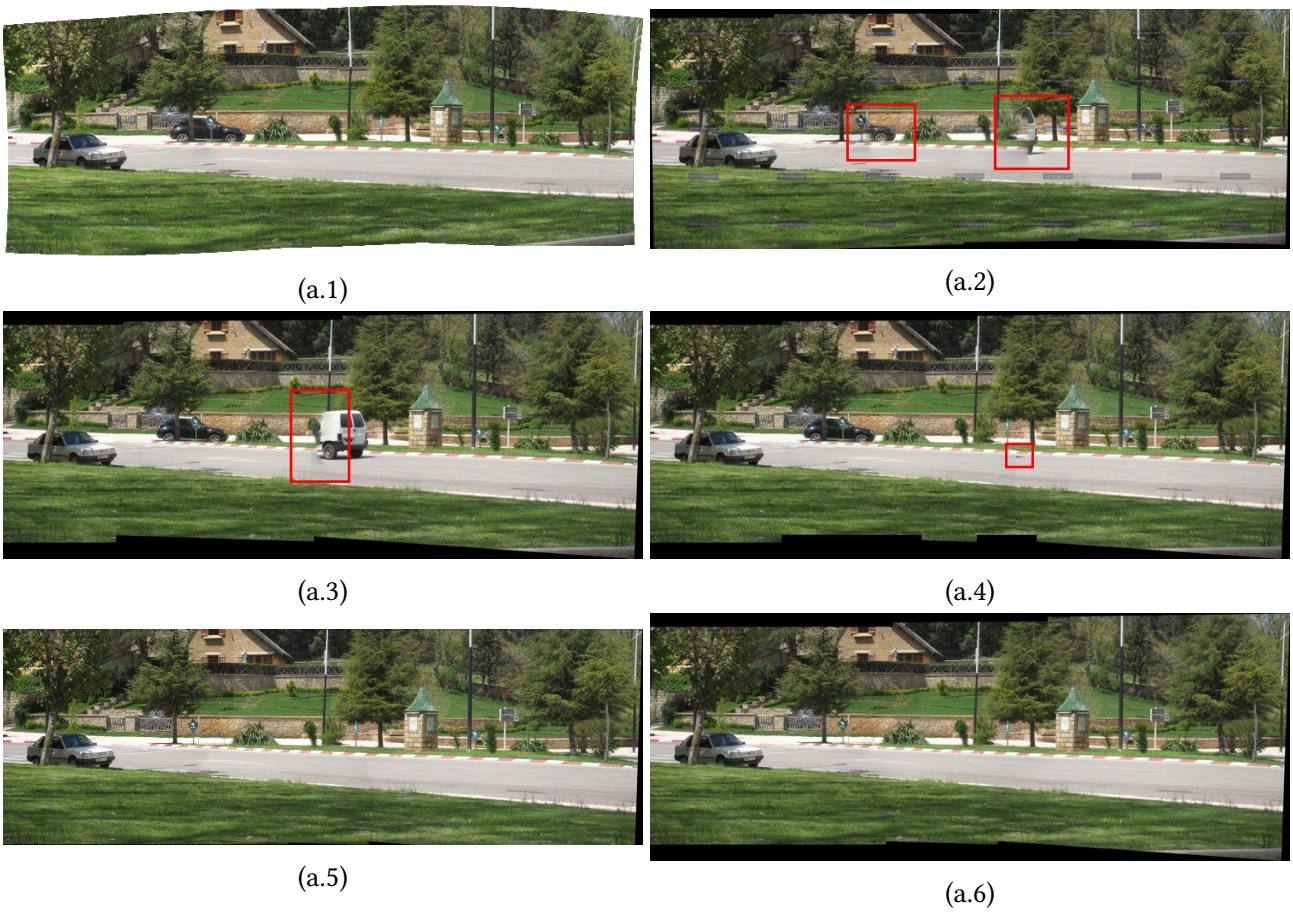
(b.4) (b.5) (b.6)

FIGURE 4.14 – Résultats mosaïques pour les séquences : (a) Bmx-trees (b) Scooter-gray ; où les logiciels et méthodes sont (1) Logiciel Hugin, (2) Logiciel PTGui, (3) méthode CSDP, (4) méthode Graphcut, (5) méthode Inlier/Outlier, (6) Méthode développée.

Pour la première séquence (Figure 4.13.a), le logiciel PTGui (Figure 4.14.a.2) et la méthode CSDP (Figure 4.14.a.3) ont créé de mauvais résultats où le ghosting est visible, tandis que le logiciel Hugin (Figure 4.14.a.1) a créé une duplication d'objets. Par contre, pour la méthode graphcut (Figure 4.14.a.4), la méthode Inlier/Outlier (Figure 4.14.a.5) et la méthode développée (Figure 4.14.a.6), les mosaïques n'ont pas créé d'artefacts. De plus, des résultats similaires ont été obtenus pour la deuxième séquence (Figure 4.13.b) où PTGui (Figure 4.14.b.2) et CSDP (Figure 4.14.b.3) ont créé un ghosting et Hugin (Figure 4.14.b.1) a généré une duplication d'objets. En plus, des méthodes Graphcut (Figure 4.14.b.4), Inlier/Outlier (Figure 4.14.b.5) et notre méthode (Figure 4.14.b.6) qui ont produit un bon résultat (Les artefacts non souhaités ont été encadrés en rouge dans les figures). En plus, les résultats visuelles sont confirmés avec les mesures quantitatives dans le tableau 4.1 où les valeurs de SSIM pour notre méthode est la plus optimale pour la première séquence, et la seconde optimale pour la deuxième

séquence, en plus d'être la plus optimale pour MS-SSIM et SSIM-Seamline.

La seconde expérimentation est effectuée sur la séquence d'images (Figure 4.13.c) qui contient différents objets dynamiques coupés qui sont présents que dans une image avec certains de ces objets qui sont incomplets. Les résultats obtenus dans la figure 4.15 montrent, d'une part, un ghosting pour PTGui (Figure 4.15.a.2) et CSDP (Figure 4.15.a.3), en plus d'une ombre qui est visible dans la méthode Graphcut (Figure 4.15.a.4), et de l'autre part, un bon résultat pour le logiciel Hugin (Figure 4.15.a.1), la méthode Inlier/Outlier (Figure 4.15.a.5) et notre méthode (Figure 4.15.a.6). Le tableau 4.1 reflète aussi ses résultats avec la méthode développée qui a les valeurs les plus élevées pour les trois mesures.



(a.1) (a.2) (a.3) (a.4) (a.5) (a.6)

FIGURE 4.15 – Résultats mosaïques pour les séquences : (a) Valley ; où les logiciels et méthodes sont (1) Logiciel Hugin, (2) Logiciel PTGui, (3) méthode CSDP, (4) méthode Graphcut, (5) méthode Inlier/Outlier, (6) Méthode développée.

La figure 4.16.a montre les résultats générés de la séquence (Figure 4.13.d) où les images d'entrées contiennent plusieurs objets dynamiques dans une scène à l'intérieur. Ces objets dynamiques ont créé beaucoup d'artefacts parce qu'ils occupent la majorité de la zone de chevauchement et laissent peu d'endroits d'arrière-plan où les "seamlines" peuvent passer. Les résultats montrent un taux de "ghosting" élevé pour Hugin (Figure 4.16.a.1), PTGui (Figure 4.16.a.2) et CSDP (Figure 4.16.a.3) et un taux de "ghosting" léger pour la méthode Inlier/Outlier (Figure 4.16.a.5). Tandis que la méthode Graphcut (Figure 4.16.a.4) et notre méthode (Figure 4.16.a.6) ont créé de bon résultats. Dans le tableau 4.1, la valeur de SSIM la plus élevée a été réalisée par la méthode Graphcut avec la méthode développée

en deuxième. Par contre, pour la valeur de MS-SSIM et SSIM-Seamline, notre méthode a produit les valeurs les plus élevées.

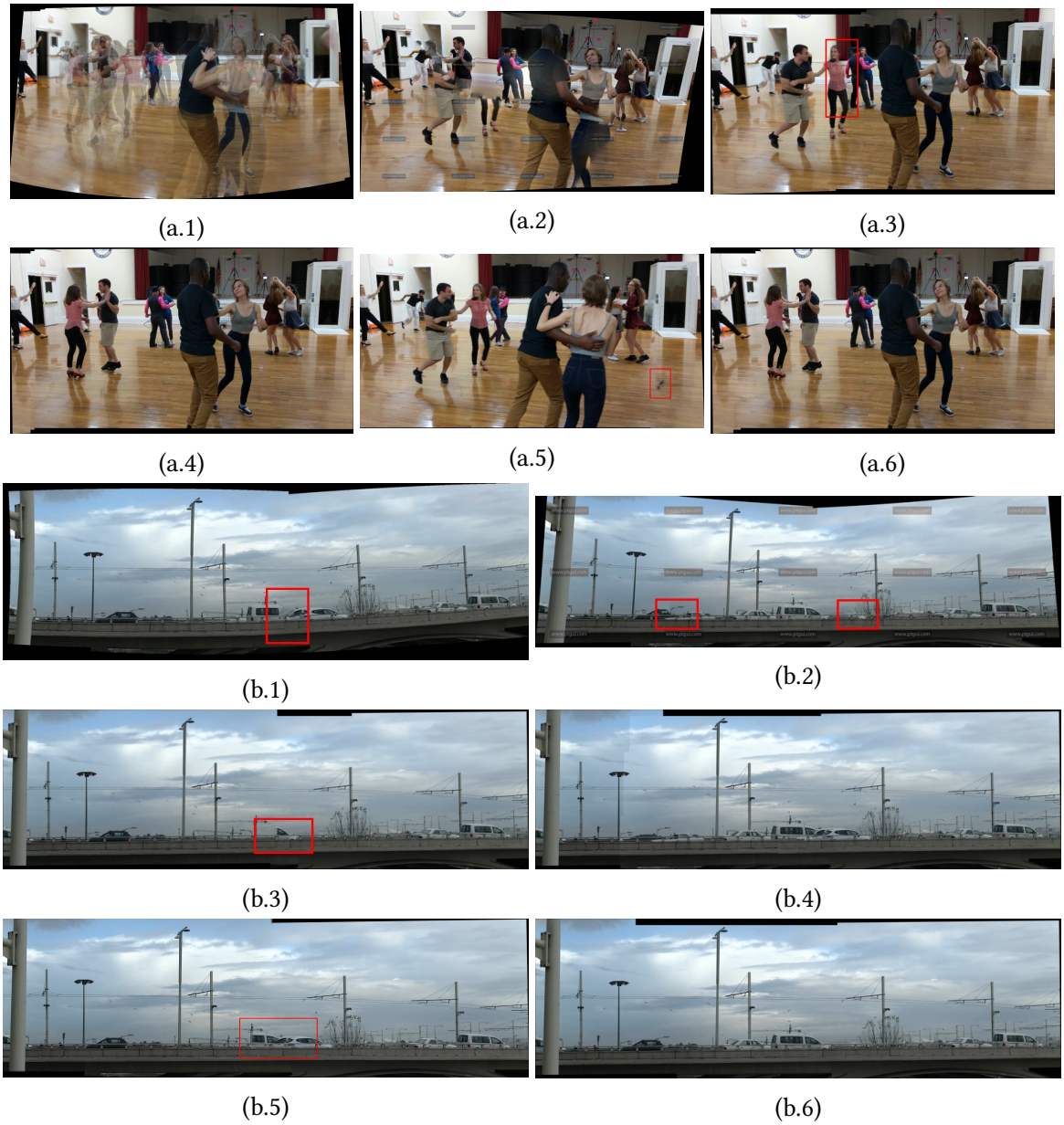


FIGURE 4.16 – Résultats mosaïques pour les séquences : (a) Lindy-hop (Fig. 12.d), (b) Bridge (Fig. 12.e) ; où les logiciels et méthodes sont (1) Logiciel Hugin, (2) Logiciel PTGui, (3) méthode CSDP, (4) méthode Graphcut, (5) méthode Inlier/Outlier, (6) Méthode développée.

Pour la dernière expérimentation, la séquence d'images (Figure 4.13.e) qui contient beaucoup d'objets et textures dynamiques a été utilisée. Les résultats dans la figure 4.16.b montre du "ghosting" avec des voitures coupées en deux pour Hugin (Figure 4.16.b.1), PTGui (Figure 4.16.b.2), CSDP (Figure 4.16.b.3) et la méthode Inlier/Outlier (Figure 4.16.b.5). En outre, de bons résultats ont été générés par la méthode Graphcut (Figure 4.16.b.4) et la méthode développée (Figure 4.16.b.6). Ces résultats peuvent être confirmés avec le tableau 4.1 où notre méthode possède la seconde valeur la plus optimale pour SSIM et les valeurs les plus optimaux pour MS-SSIM et SSIM-Seamline.

	CSDP	Graphcut	In_Out	Notre méthode	CSDP	Graphcut	In_Out	Notre méthode
	SSIM				MS_SSIM			
Fig. 4.13.a	0, 6273	0, 6405	0, 6308	**0,7026**	0, 9605	0, 9901	0, 9681	**0,9936**
Fig. 4.13.b	0, 7379	0, 7375	**0,7436**	0, 7414	0, 9717	0, 9773	0, 9660	**0,9877**
Fig. 4.13.c	0, 6560	0, 6837	0, 6612	**0,6905**	0, 9435	0, 9517	0, 9628	**0,9898**
Fig. 4.13.d	0, 7644	**0,7779**	0, 7705	0, 7761	0, 8811	0, 9499	0, 9760	**0,9894**
Fig. 4.13.e	0, 8007	**0,8140**	0, 7362	0, 8127	0, 9643	0, 9843	0, 9495	**0,9972**

	CSDP	Graphcut	In_Out	Notre méthode
	SSIM_Seamline			
Fig. 4.13.a	0, 8970	0, 9745	0, 9030	**0,9920**
Fig. 4.13.b	0, 7666	0, 9684	0, 8834	**0,9860**
Fig. 4.13.c	0, 8565	0, 8519	0, 9116	**0,9439**
Fig. 4.13.d	0, 8498	0, 9547	0, 9411	**0,9860**
Fig. 4.13.e	0, 9809	0, 9840	0, 9131	**0,9915**

TABLE 4.1 – Valeurs de SSIM, MS_SSIM, SSIM_Seamline pour les méthodes CSDP, Graphcut, Inlier/Outlier et notre méthode pour les mosaïques obtenues à partir des images de la figure 4.13. (Valeurs optimaux en gras)

Finalement, en plus des comparaisons de la table 4.1, la figure 4.17 montre le temps d'exécution des méthodes CSDP, Graphcut, Inlier/Outlier et notre méthode. Ces temps d'exécutions sont calculés avec une moyenne des temps obtenus après 10 exécutions des algorithmes.

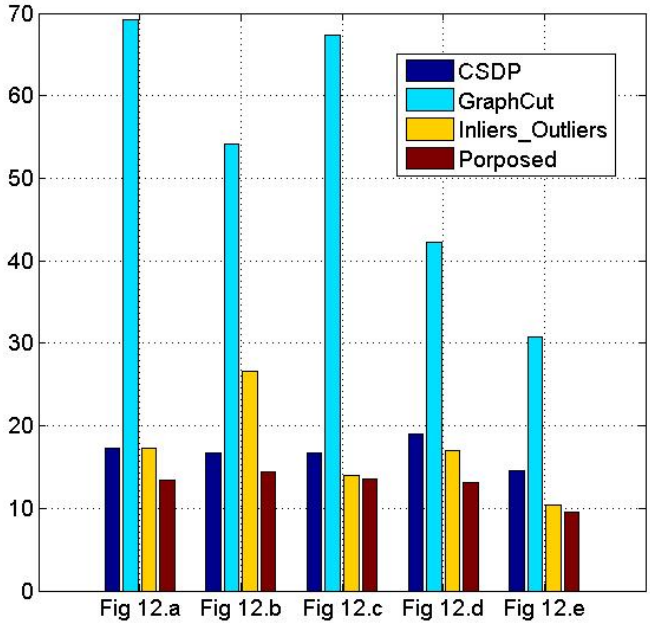


FIGURE 4.17 – Temps d'exécution des différentes approches pour toutes les séquences en secondes.

Notre méthode, la méthode CSDP et la méthode Inlier/Outlier produisent tous un temps presque similaire sauf la méthode Graphcut qui a un temps élevé. En effet, la méthode développée est légèrement plus rapide que les autres méthodes avec un temps presque similaire et en même temps elle

crée de meilleurs résultats. Et même si la méthode Graphcut a créé que des bons résultats, la méthode développée reste plus rapide.

4.4.3 Analyses des mesures obtenues

Pour résumer les résultats du tableau 4.1, nous notons que notre méthode a les valeurs les plus élevées pour les mesures MS-SSIM et SSIM-Seamline, tandis qu'elle est la deuxième optimale dans certains cas pour la mesure SSIM. Ainsi, on peut déduire qu'elle a calculé la meilleure "seamline" par rapport aux autres méthodes. Effectivement, en moyenne, à partir des mesures quantitatives calculées à partir de la méthode développée, nous avons réalisé :

- Pour **SSIM** : une amélioration de +4,15% par rapport à la méthode CSDP de Zeng et al., de +2,32% par rapport à la méthode Graphcut de Li et al. et de +5,44% par rapport à la méthode basée sur le détecteur de contours.
- Pour **MS-SSIM** : une amélioration de +5,14% par rapport à la méthode CSDP de Zeng et al., de +0,35% par rapport à la méthode Graphcut de Li et al. et de +2,81% par rapport à la méthode basée sur le détecteur de contours.
- Pour **SSIM-Seamline** : une amélioration de +13,30% par rapport à la méthode CSDP de Zeng et al., de +3,69% par rapport à la méthode Graphcut de Li et al. et de +7,67% par rapport à la méthode basée sur le détecteur de contours.

Au niveau des temps d'exécutions, en moyenne la méthode développée a realisé un gain de temps de +24,12% par rapport à la méthode CSDP de Zeng et al., de +74,38% par rapport à la méthode Graphcut de Li et al. et de +20,50% par rapport à la méthode basée sur le détecteur de contours.

Ainsi à partir du tableau 4.1 et la figure 4.17 et les différents résultats mosaïques obtenus, nous pouvons conclure que la méthode basée sur un mouvement fractionnaire Brownien et l'algorithme A* génère de meilleurs résultats.

4.5 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode de la catégorie des méthodes basées sur le calcul d'une "seamline" qui utilise un mouvement fractionnaire (fBm) modifié avec la fonction de similarité "Zero Normalized Cross Correlation" (ZNCC) au lieu d'une fonction de bruit pour créer la carte d'énergie. Puis, nous avons calculé la "seamline" avec l'algorithme d'apprentissage automatique A*. En outre, en combinant entre la carte d'énergie et l'algorithme A*, on a créé une "seamline" flexible qui évite les objets dynamiques ce qui a été prouvé à travers les différents tests effectués. Les motivations initiales dans la création de cette méthode étaient d'obtenir une bonne détection des objets dynamiques et des erreurs de projection pour introduire un algorithme d'apprentissage automatique qui va générer une bonne "seamline". Ces motivations ont été réalisées. Finalement, en analysant les différents résultats en modifiant le nombre de distributions et la fréquence, nous avons déduit les meilleurs paramètres pour générer une mosaïque de haute qualité.

Chapitre **5**

Mosaïque d'images basée sur un système de recommandation.

5.1 Introduction 96
5.2 Segmentation des images avec un "clustering" 97
5.2.1 Recalage d'images 97
5.2.2 "Clustering" K-means 99
5.2.3 Catégorisation des pixels 100
5.3 Création d'une mosaïque avec un système de recommandation 103
5.3.1 Création des profils des pixels 103
5.3.2 Recommandation de la mosaïque 104
5.4 Résultats expérimentaux 106
5.4.1 Résultats 109
5.4.2 Analyses des mesures obtenues 113
5.5 Conclusion 114

5.1 Introduction

Dans le chapitre 3 et le chapitre 4, nous avons présenté des méthodes basées sur le calcul d'une "seamline". Dans le présent chapitre, nous allons développer une méthode basée sur la reconstruction de régions. Cette dernière est basée sur un système de recommandation "item-to-item" qui va générer la mosaïque finale. En effet, ce que nous voulons réaliser c'est :

- Premièrement, créer une méthode de mosaïque d'images qui utilise toutes les images simultanément pour la génération de la mosaïque finale. En effet, comme nous l'avons remarqué dans le chapitre 2, les méthodes basées sur le calcul d'une "seamline" utilisent deux images à la fois tandis que la majorité des méthodes basées sur la reconstruction d'images utilisent trois images au maximum ;
- Deuxièmement, générer une mosaïque sans la création d'artefacts ;
- Finalement, introduire un système de recommandation pour obtenir une mosaïque de haute qualité avec un gain de temps.

En vue de ses motivations, les contributions que nous avons réalisé sont les suivantes :

- La première contribution consiste à utiliser toutes les images d'entrées **simultanément** après qu'elles aient subi un recalage d'images. L'objectif de cette opération est de détecter tous les objets dynamiques en une seule étape pour obtenir un gain significatif en temps d'exécution. En outre, en utilisant toutes les images simultanément, nous pouvons utiliser un algorithme de "clustering". Par conséquent, l'algorithme k-means a été utilisé pour segmenter les pixels de toutes les images projetées dans différentes classes ("clusters"). Chaque "cluster" contiendra l'information des pixels similaires de toutes les images. Ainsi, c'est possible de comparer la position d'un pixel précis dans chaque image et observer si les pixels ayant la même position dans toutes les images appartiennent au même "cluster" ou pas.
- Dans la seconde contribution, nous avons localisé la position des pixels qui peuvent créer des artefacts avec précision en utilisant des nouveaux "clusters" créés à partir des classes k-means. En effet, chaque classe k-means, peut être divisé en deux classes : un "cluster" qui contient des pixels dont la position existe dans toutes les images et un "cluster" pour les pixels qui n'existent pas dans toutes les images. Ainsi, en combinant les pixels du deuxième "cluster" de chaque classe k-means, on obtient la position des objets dynamiques. De plus, ces nouveaux "clusters" vont aussi définir les catégories où se trouve chaque pixel. Ces catégories vont être utilisées plus tard pour effectuer la recommandation.
- Finalement, en considérant que chaque image est un utilisateur, chaque pixel comme un objet et l'interprétation de chaque pixel comme un profil d'objet, un système de recommandation peut être utilisé pour recommander le meilleur résultat possible pour générer une mosaïque. En effet, chaque pixel fait partie d'une catégorie et d'une zone de chevauchement d'un certain

nombre d'images, ce qui permet de créer un profil d'objet pour chaque pixel où ce profil dépend de :

(i) Premièrement, un score créé avec les valeurs d'intensités dans chaque canal après réajustement pour que des valeurs élevées soient interprétées comme un score élevé ;

(ii) Deuxièmement, la valeur de texture de chaque canal obtenue avec un filtre Sobel dans les deux directions ;

(iii) Finalement, l'amplitude et l'orientation de l'objet dans chaque canal.

Ainsi, tous les profils sont utilisés avec la méthode du filtre collaborative du système de recommandation "item-to-item" pour choisir les meilleurs pixels pour la mosaïque finale.

La figure 5.1 est une généralisation des étapes de la méthode développée dans ce chapitre. Premièrement, les images d'entrées sont projetées dans le même repère. Ensuite, parallèlement, les scores des pixels sont calculés et les pixels sont catégorisés dans des clusters qui ont été obtenus avec les clusters k-means et leur zones de chevauchements. Finalement, en combinant entre les scores et la segmentation, une recommandation est effectuée pour créer une image mosaïque.

Après cette introduction dans la première section, le plan du chapitre est comme suit : la seconde section va définir comment chaque pixel de chaque image est catégorisé selon sa position et l'algorithme k-means, puis elle explique comment la segmentation est effectuée avec les k-"clusters". Ensuite, la troisième section formule la méthode de calcul des profils d'objets de chaque pixel et montre comment la recommandation est effectuée. Puis, les résultats expérimentaux sont discutés dans la quatrième section et finalement le chapitre se termine avec une conclusion.

5.2 Segmentation des images avec un "clustering"

Avant d'utiliser un système de recommandation, il faut d'abord classifier les pixels dans différentes catégories. Ces catégories vont définir quels sont les pixels similaires qui vont éventuellement avoir la même recommandation. Ce qui fait que la catégorie d'un pixel est un attribut des profils d'objets. Cependant, même si les pixels d'une catégorie auront la même recommandation, chaque pixel aura son propre profil d'objet qui sera unique.

Dans cette section, nous allons présenter les algorithmes de la phase du recalage. Puis, la procédure de catégorisation de chaque pixel en utilisant leur position dans chaque image et l'algorithme k-means.

5.2.1 Recalage d'images

Les algorithmes utilisés pour effectuer le recalage d'images sont SIFT [24] (Section 1.3.4.1) pour la détection et la description des points d'intérêts et RANSAC [38] (Section 1.3.4.2) pour la mise en correspondance et le calcul de l'homographie.

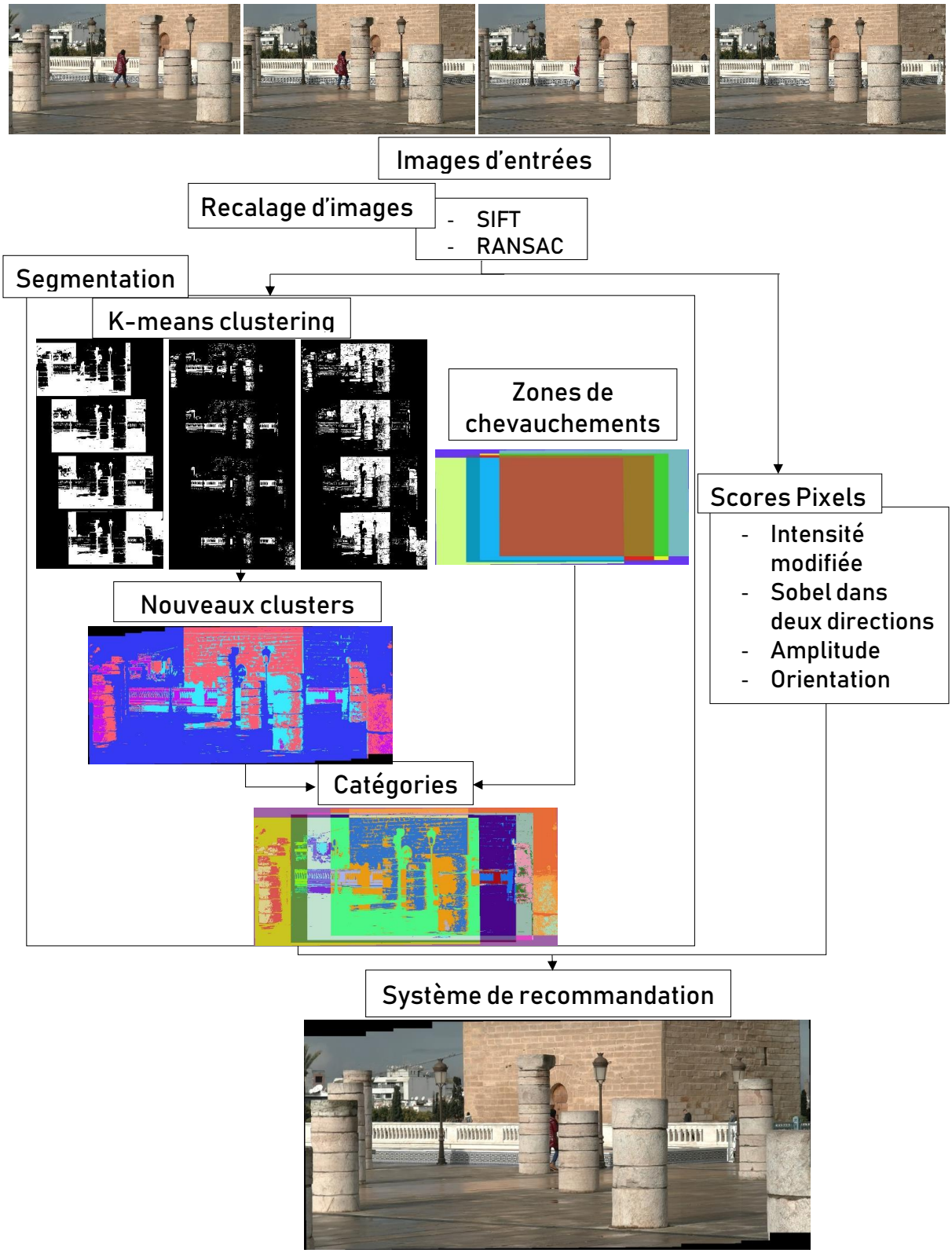
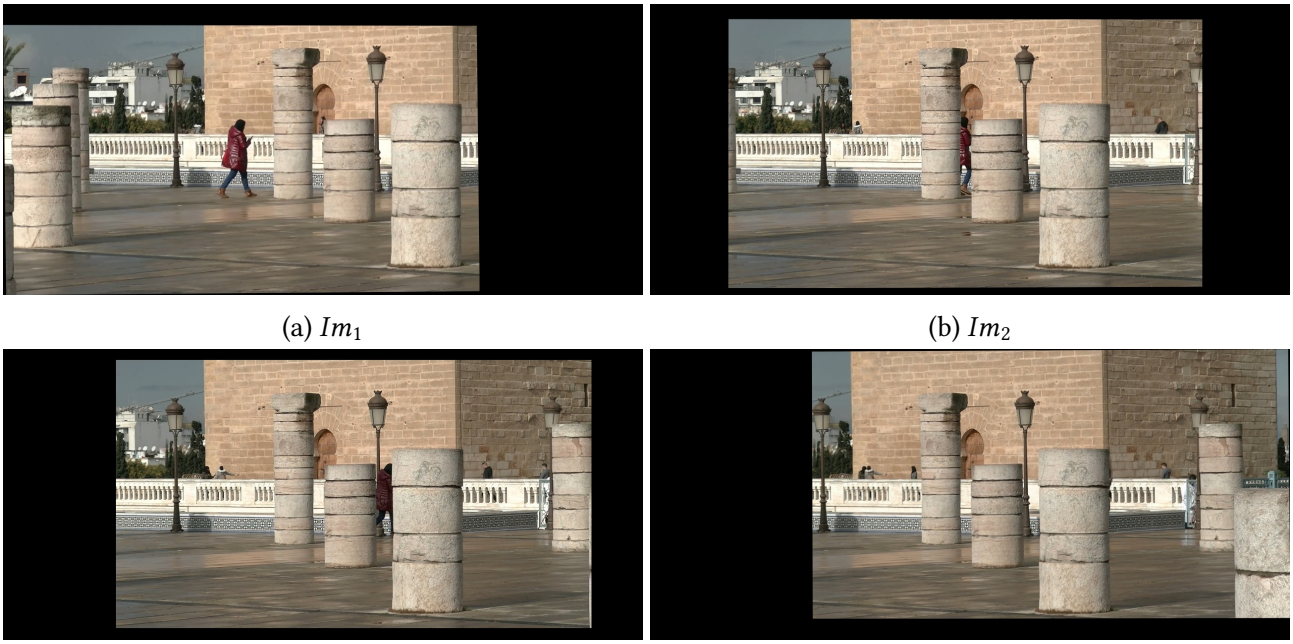


FIGURE 5.1 – Etapes pour la recommandation d'une mosaïque avec plusieurs images d'entrées.

Soit Im_i des images d'entrées qui vont être utilisées pour expliquer la méthode. La figure 5.2 montre ces images après leur projection dans le même système de coordonnées.



(a) Im_1 (b) Im_2

(c) Im_3 (d) Im_4

FIGURE 5.2 – Images d'entrées après le recalage d'images.

5.2.2 "Clustering" K-means

Dans le but de créer une mosaïque avec un système de recommandation, les pixels de chaque image vont être regroupés dans différentes catégories. Ces catégories vont permettre d'attribuer la même recommandation aux pixels qu'elles incluent ce qui va accélérer l'algorithme.

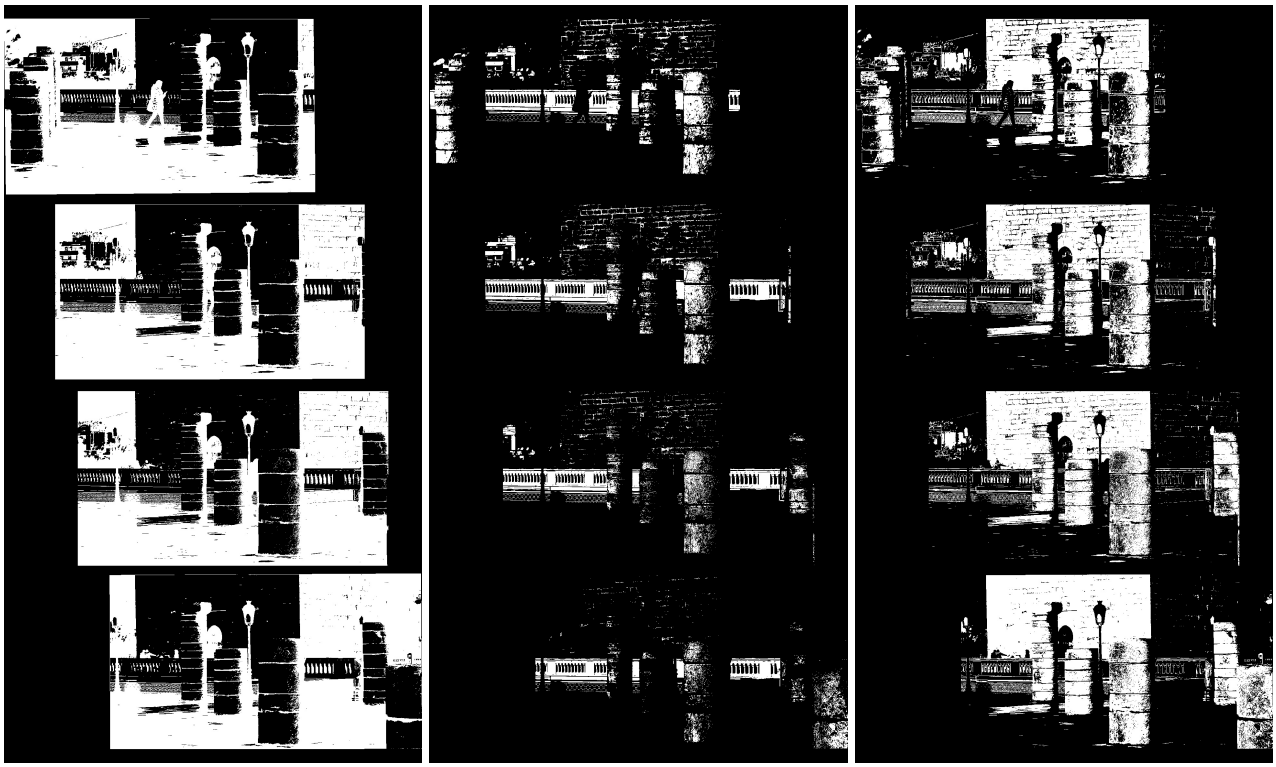
Vu qu'à chaque nouvelle séquence d'images que nous voulons utiliser pour créer une mosaïque des images d'entraînements n'existent pas, les pixels vont être regroupés en utilisant un algorithme qui n'as pas besoin de données d'entraînements, l'algorithme de "clustering" k-means [84]. Ce dernier va segmenter les pixels similaires en se basant sur leurs valeurs dans des clusters. Effectivement, les valeurs utilisées dans l'algorithme k-means sont les valeurs de tous les pixels visibles de toutes les images d'entrées après leur projection dans le même repère. De plus, les pixels hors bordures qui apparaissent après la projection ne sont pas pris en compte pour éviter la création de "clusters" incorrects. Sur la figure 5.2, les pixels hors bordures sont les pixels noires qui ne font pas partie de l'image. Dans le système de recommandation, les pixels hors bordures sont considérés comme des objets appartenant à un utilisateur spécifique (ici c'est l'image) qui ne possèdent pas de profils (valeur d'intensité).

L'algorithme k-means partitionne les données en k classes en attribuant à chaque classe un centroïde, puis regroupe les points de données d'entrées à la classe la plus proche en minimisant la dis-

tance au carré entre le centroïde et chaque point de données. Les étapes de l'algorithme sont les suivantes :

1) L'algorithme commence en choisissant k centroïdes aléatoires parmi les points de données d'entrées.
2) Une distance est calculée entre tous les points de données et les k clusters. Puis les points de données sont affectés à la classe la plus proche.
3) De nouveaux centroïdes sont calculés en utilisant les points de données de chaque classe (Ce n'est pas obligatoire que les centroïdes fassent partie des points de données).
4) Si la différence entre les nouveaux centroïdes et les anciens centroïdes est inférieure à une valeur ε l'algorithme s'arrête. Sinon, l'étape 2,3 et 4 se répète en utilisant les nouveaux centroïdes jusqu'à convergence ou jusqu'à arriver à un nombre d'itérations choisi par l'utilisateur.

En utilisant les images d'entrées de la figure 5.2, la figure 5.3 montre $k = 3$ classes obtenues après 3 itérations. Dans cet exemple, k-means a été calculé en utilisant les valeurs des pixels de toutes les images. Chaque classe Cl obtenu par k-means contient les pixels de toutes les images.



Classe Cl_1 Classe Cl_2 Classe Cl_3

FIGURE 5.3 – Classes Cl obtenues des images de la figure 5.2 avec l'algorithme k-means, $k = 3$.

5.2.3 Catégorisation des pixels

Après la classification des pixels en différents clusters (Figure 5.3), la prochaine étape est la création de catégories pour les pixels.

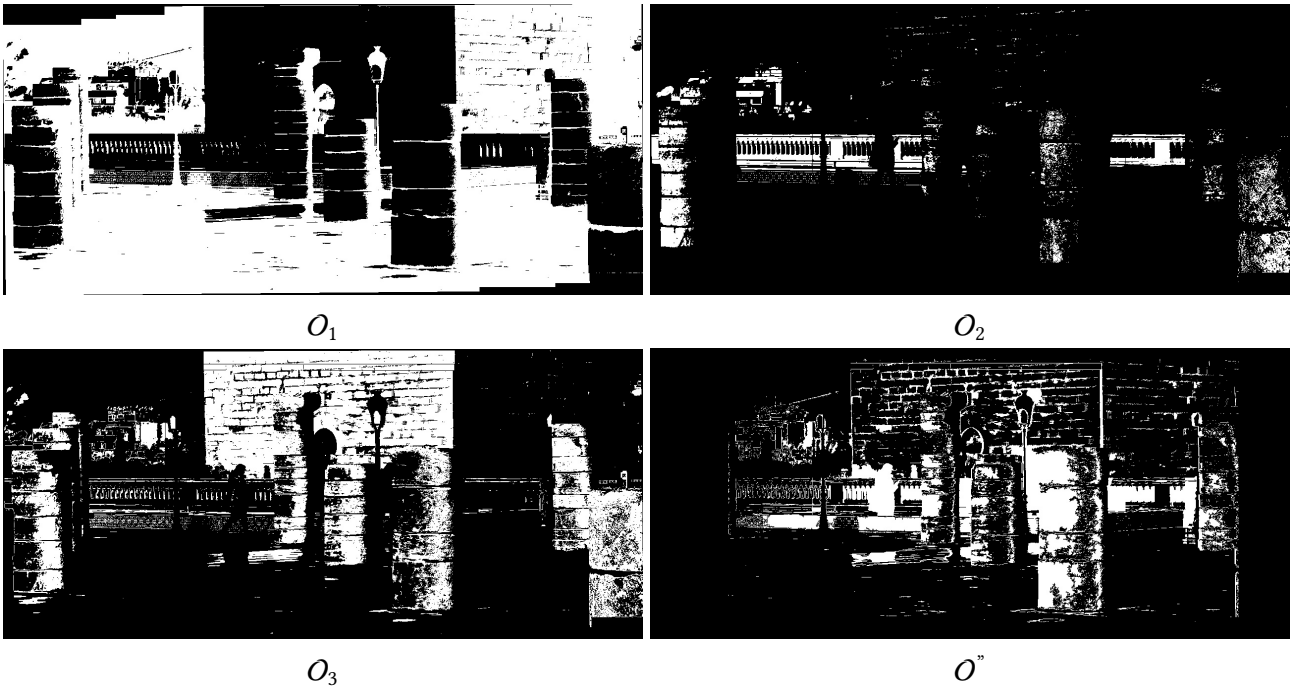
Dans un système de recommandation, les catégories sont considérées comme un ensemble d'objets qui ont les mêmes attributs, ce qui implique qu'ils auront des recommandations similaires. Dans notre cas, les pixels du même "cluster" sont considérés comme appartenant à la même catégorie. Cependant, avec l'existence des objets dynamiques, et les différentes zones de chevauchements, des ajustements doivent être effectués :

- Premièrement dans chaque Cl_q avec $q = 1...k$, chaque image Im_i est comparée avec l'image Im_{i+1} du même cluster. Effectivement, en comparant les images ainsi, chaque "cluster" peut être séparé en deux régions :
  - O_q qui contient les coordonnées des pixels qui font partie du "cluster" dans toutes les images.
  - $O_q^{'}$ qui contient les coordonnées des pixels qui n'existent que dans certaines images du "cluster".

  Les pixels qui sont comparés avec des pixels hors bordures sont considérés comme faisant partie de O_q. Les figures 5.4.a, 5.4.b et 5.4.c montrent les régions similaires obtenus par O_q en utilisant les "clusters" de la figure 5.3. De plus, l'équation 5.1 défini le premier ajustement.

$$\begin{cases} O_q = \{(x,y)/\forall i = 0...N-1 (x,y) \in Cl_q(Im_i)\} \\ O_q^{'} = \{(x,y)/\exists i = 0...N-1 (x,y) \notin Cl_q(Im_i)\} \end{cases} \tag{5.1}$$

  Où :
  - x et y sont les coordonnées des pixels dans les images Im_i.
  - q est l'index du "cluster" avec $q = 1...k$.
  - N le nombre d'images.
  - $Cl_q(Im_i)$ les pixels de l'image Im_i qui font partie du "cluster" Cl_q.



O_1 O_2

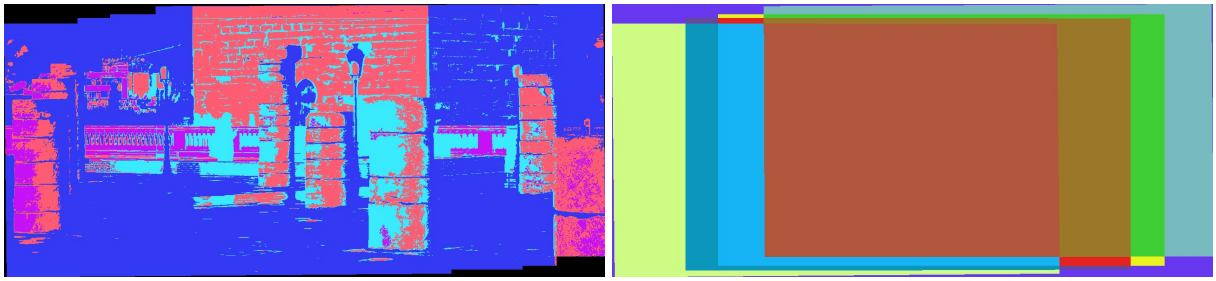
O_3 $O^{"}$

FIGURE 5.4 – Zones similaires O_q obtenues de chaque "cluster" Cl_q de la Figure 5.3 et les zones non similaires détectées $O^{"}$ qui peuvent causer des artefacts.

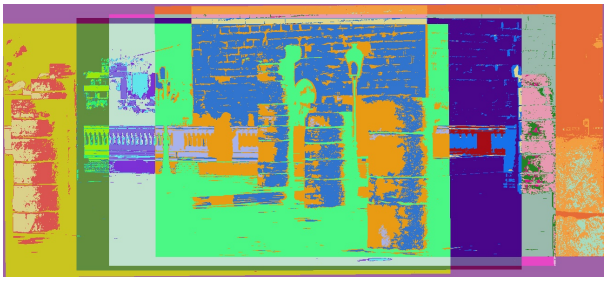
- Deuxièmement, pour éviter la création de duplication d'objets, tous les pixels qui ont été catégorisés comme pixels dynamiques dans O'_q sont fusionnés en une seule catégorie $O^{"}$ par l'équation 5.2. La figure 5.4.d montre la position des pixels dynamiques de la catégorie $O^{"}$ qui peuvent créer des artefacts avec la fusion des O'_q. Ainsi, jusqu'à présent le nombre de catégories créées est de $k+1$ catégories. En effet, k catégories de O_q avec $q = 1...k$ et une catégorie $O^{"}$ qui contient tous les objets dynamiques. En outre, afin d'avoir une meilleure précision dans $O^{"}$, les trous qui existent à l'intérieur des pixels dynamiques sont remplis, ce qui implique que les pixels qui constituent les trous sont enlevés des catégories O_q et ajoutés à $O^{"}$. Figure 5.5.a montre la combinaison des catégories O_q et $O^{"}$ dans une seule figure pour créer une segmentation de toutes les images.

$$O^{"} = \{O'_0 \cup O'_1 \cup ... \cup O'_k\} \tag{5.2}$$

- Finalement, segmenter les pixels en utilisant plusieurs catégories créées à partir des "clusters" k-means n'est pas suffisant parce qu'il existe plusieurs zones de chevauchements avec un nombre différents d'images. En effet, dans un système de recommandation, chaque pixel possède un profil, et puisque les pixels qui n'ont pas de profils (les pixels hors bordures d'une image) ne vont pas être utilisés, alors chaque catégorie est divisée selon le nombre d'images présentes dans la zone de chevauchement. Ainsi, chaque zone de chevauchement des images (Figure 5.5.b) aura $k+1$ catégories. Ce qui fait que le nombre total de catégories dans l'ensemble C, l'ensemble qui contient tous les clusters, est $n_{overlap} * (k+1)$ avec $n_{overlap}$ le nombre de zones de chevauchements. La figure 5.5.b contient les zones de chevauchements des images d'entrées de la figure 5.2 et la figure 5.5.c montre la combinaison entre les zones de chevauchement et la segmentation où chaque zone différente est représentée par une couleur.



(a) (b)



(c) C

FIGURE 5.5 – Segmentation finale (a) Catégories obtenues en combinant toutes les zones de la figure 5.4, (b) Zones de chevauchements (c) L'ensemble C qui contient la combinaison de toutes les catégories et les zones de chevauchements (Figure 5.5.a et 5.5.b).

5.3 Création d'une mosaïque avec un système de recommandation

Pour comprendre comment le système de recommandation va fonctionner, nous allons premièrement définir comment chaque élément des mosaïques d'images peut être interprété dans le système. En effet, chaque image d'entrée après la phase de recalage est considérée comme un utilisateur. Vu que chaque image contient des pixels, alors les pixels seront les objets qu'un utilisateur a scoré. Ainsi, la mosaïque résultante est considérée comme un nouvel utilisateur dont les pixels vont être recommandés en utilisant les pixels des utilisateurs déjà existants.

Pour pouvoir réaliser la recommandation, chaque objet (pixel) a besoin d'un profil. Ce profil est un vecteur qui contient plusieurs informations de l'objet : ses valeurs d'intensités, sa position, son amplitude, son orientation et sa catégorie. Ces profils sont ensuite utilisés pour créer une matrice de comparaison basée sur la méthode du "item-to-item" filtre collaboratif où pour chaque objet spécifique des objets similaires sont trouvés. C'est ainsi que la mosaïque résultante va être remplie de pixels estimés par les scores des pixels similaires.

Dans cette section, nous allons premièrement définir comment créer le profil de chaque pixel, puis nous présenterons la méthode de génération de la mosaïque avec le système de recommandation.

5.3.1 Création des profils des pixels

Un profil d'objet est un ensemble de différents attributs qui vont caractériser chaque objet pour qu'il puisse être comparé dans le système de recommandation. Nous avons défini les différents attributs d'un profil $Profile_i(x, y)$ d'une image Im_i et d'un pixel (x, y) de la manière suivante :

- Un score qui est obtenu avec les valeurs d'intensités d'une image suivant la position du pixel. Ce score est calculé avec l'équation 5.3. En effet, puisque les valeurs d'intensités de chaque canal ne sont pas suffisantes pour déterminer si un pixel a un bon score ou pas, de nouvelles valeurs sont calculées pour qu'une valeur élevée indique un bon score. Tout d'abord, une moyenne pour chaque pixel est calculée avec les valeurs des pixels ayant la même position dans toutes les images, ensuite une soustraction est effectuée entre cette moyenne et les pixels associés à la moyenne. Quand le pixel est proche de la moyenne, la soustraction est proche de 0 ainsi il est considéré comme un pixel de bonne qualité. Cette valeur est ensuite réajustée pour qu'une valeur élevée signifie un bon score.

$$Score_i(x, y) = 254 - |Im_i(x, y) - \mu(x, y)| \tag{5.3}$$

Où :

- x et y sont les coordonnées du pixel des différentes images.
- i est l'indice de l'image actuelle avec $i = 0...N - 1$.

  - $\mu(x, y)$ la moyenne des valeurs du pixel (x, y) de toutes les images avec $\mu(x, y) = \frac{1}{a} \sum_{i=0}^{N-1} p * Im_i(x, y)$ où $p = 0$ si le pixel est hors bordures sinon $p = 1$ et a le nombre d'images où les pixels (x, y) ne sont pas hors bordures.

- La texture de chaque canal. Elle est calculée avec le filtre Sobel dans deux directions, horizontale et verticale pour obtenir Gx_i et Gy_i pour chaque image Im_i afin de discerner où les différences de textures existent horizontalement et verticalement.

- L'amplitude de chaque pixel G_i qui est calculée par l'équation 5.4 pour chaque canal de chaque image Im_i. En effet, l'amplitude amplifie les valeurs pour montrer précisément où les intensités des images changent. De petites amplitudes indiquent que la texture dans cette région est uniforme tandis que de grandes amplitudes indiquent que la région fait partie des contours de l'image.

$$G_i(x, y) = \sqrt{Gx_i(x, y)^2 + Gy_i(x, y)^2} \tag{5.4}$$

- L'orientation θ_i calculée par l'équation 5.5 pour montrer la direction des changements d'intensités dans le voisinage d'un pixel. La valeur est en degré dans l'intervalle $[0, 360]$. Quand la valeur est petite alors il n'y a pas beaucoup de changement dans le voisinage.

$$\theta_i(x, y) = arctan \frac{Gy_i(x, y)}{Gx_i(x, y)} \tag{5.5}$$

Pour chaque pixel, le profil est défini comme un vecteur formulé par l'équation 5.6. Chaque vecteur contient les cinq attributs décrit précédemment dans chaque canal. Ainsi, le vecteur possède 15 éléments.

$$Profile_i = (Score_i, Gx_i, Gy_i, G_i, \theta_i)) \tag{5.6}$$

5.3.2 Recommandation de la mosaïque

Dans le but de créer la mosaïque finale, nous allons utiliser les profils de chaque pixel $Profile_i$ et les catégories C. En effet, la mosaïque est considérée comme un nouveau utilisateur où chaque objet (pixel) va être recommandé à l'utilisateur. Ces objets sont recommandés par catégorie, ainsi pour chaque catégorie une recommandation est calculée.

Chaque catégorie dans C (Figure 5.5.c) contient un certain nombre de pixel, et chaque pixel possède un profil dans chaque image. Ainsi, la recommandation est calculée en utilisant une mesure de similarité sur tous les pixels d'une catégorie. Cette mesure de similarité va comparer les pixels disponibles qui ne sont pas hors bordures de toutes les images et le résultat va être l'indice i de l'image Im_i qui va être recommandé comme résultat pour la catégorie.

Ainsi, pour chaque catégorie $c \in C$:

- Les vecteurs $\vec{V_i}$ sont créés en concaténant tous les profils des pixels dans la catégorie c. L'équation 5.7 défini ce vecteur. En outre, les pixels hors bordures ne sont pas inclus. Dans cette situation, i va varier entre 0 et $a - 1$ avec a, comme défini précédemment, est le nombre d'images

ayant un pixel disponible dans la zone de chevauchement de la catégorie. Chaque vecteur $\vec{V_i}$ est associé à son image correspondante Im_j où j est l'indice de l'image associée au vecteur $\vec{V_i}$.

$$\vec{V_i} = (Profile_i(0), ..., Profile_i(P-1)) \tag{5.7}$$

Où P est le nombre de pixels dans la catégorie c.

- Chaque vecteur $\vec{V_i}$ est comparé avec les autres vecteurs en utilisant la mesure de similarité, la similarité cosinus, souvent utilisée dans les systèmes de recommandations. Elle est calculée par l'équation 5.8. La valeur obtenue de $SimCos$ est entre $[0, 1]$ où une valeur proche de 1 signifie que les vecteurs sont similaires.

$$SimCos(\vec{V_m}, \vec{V_n}) = \frac{\vec{V_m}.\vec{V_n}}{||\vec{V_m}|| * ||\vec{V_n}||} \tag{5.8}$$

Avec $SimCos(\vec{V_m}, \vec{V_m}) = 1$

- La matrice de comparaison M qui contient tous les vecteurs $\vec{V_i}$ est créée par l'équation 5.9.

$$M = \begin{bmatrix} SimCos(\vec{V_0}, \vec{V_0}) & \dots & SimCos(\vec{V_0}, \vec{V_{a-1}}) \\ SimCos(\vec{V_1}, \vec{V_0}) & \dots & SimCos(\vec{V_1}, \vec{V_{a-1}}) \\ \vdots & \ddots & \vdots \\ SimCos(\vec{V_{a-1}}, \vec{V_0}) & \dots & SimCos(\vec{V_{a-1}}, \vec{V_{a-1}}) \end{bmatrix} \tag{5.9}$$

M est une matrice symétrique puisque $SimCos(\vec{V_m}, \vec{V_n}) = SimCos(\vec{V_n}, \vec{V_m})$.

- En utilisant soit la matrice triangulaire supérieure ou la matrice triangulaire inférieure de M, la recommandation va être l'image Im_j d'indice j correspondante au vecteur $\vec{V_i}$ dans la ligne avec le score le plus élevé. Ensuite, tous les pixels de la catégorie sont attribués à la mosaïque finale à partir de l'image Im_j. La figure 5.6 montre le résultat obtenu des images de la figure 5.2 en utilisant les catégories de la figure 5.5.c.

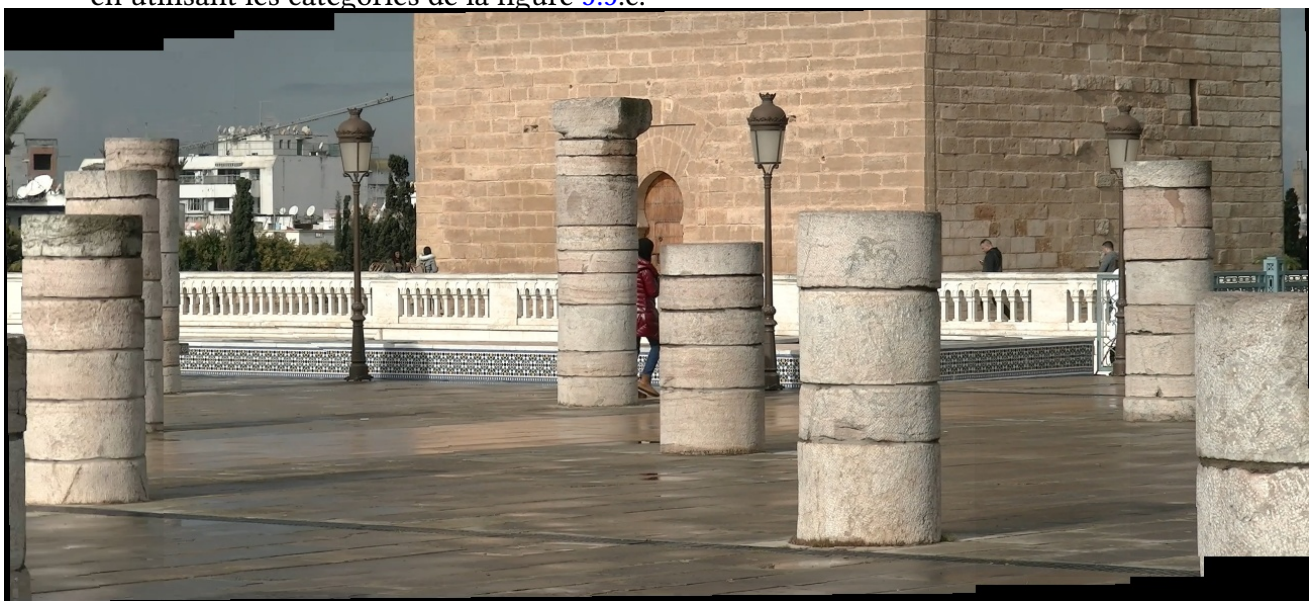


FIGURE 5.6 – Résultat mosaïque des images de la figure 5.2.

La figure 5.7 résume la procédure de la création d'une mosaïque avec un système de recommandation.

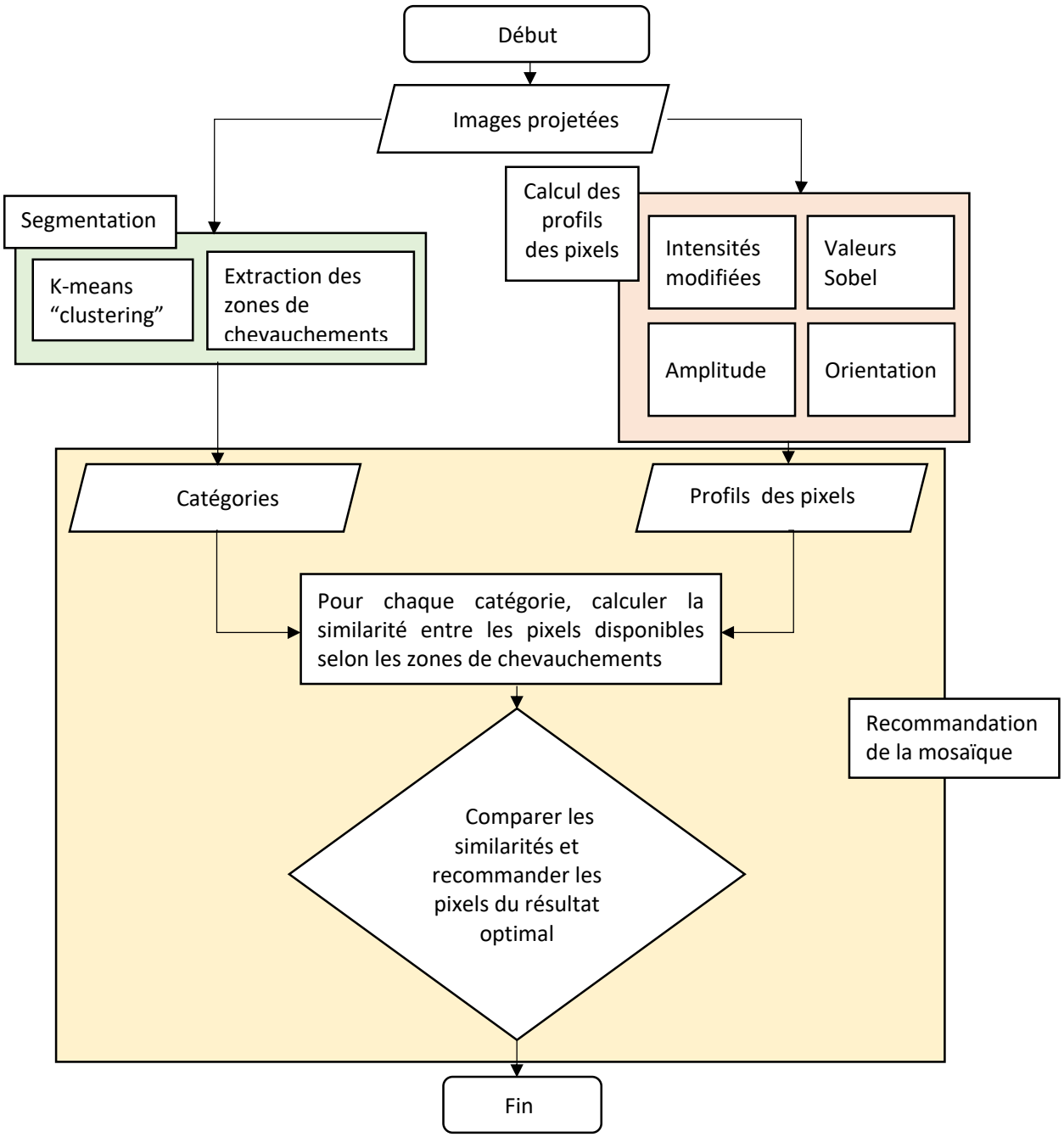


FIGURE 5.7 – Algorithme de la méthode développée basée sur un système de recommandation.

5.4 Résultats expérimentaux

Dans les mosaïques d'images dynamiques, un mouvement peut prendre plusieurs formes, ça peut être un objet dynamique, un mouvement de la caméra, un zoom, une combinaison de ces derniers etc... Pour montrer l'efficacité du système de recommandation proposé, plusieurs exemples vont être utilisés pour tester différentes situations et pour montrer les résultats recommandés. Les figures 5.8, 5.9, 5.10, 5.11 et 5.12 montrent les différentes séquences d'images utilisées pour effectuer les tests. Les deux premières séquences ont été capturées avec un JVC Camcorder GC-PX100BE sans utiliser un trépied (La figure 5.2 a été capturée de la même manière) et les trois dernières séquences ont été prises de

vidéos de Youtube. Le choix de pas utiliser un trépied et l'utilisation des vidéos de Youtube est simple, c'est tout d'abord le moyen le plus efficace pour tester l'algorithme sans utiliser des ajustements pour faciliter la génération de résultats de qualité. Ensuite, c'est aussi le moyen le plus proche pour simuler l'utilisation de l'algorithme dans des situations réelles.

Parmi les défis rencontrés dans la création de mosaïques, il y a :

- La présence d'occlusion : quand un objet dynamique se déplace derrière un élément statique et n'est plus visible pour une période de temps ou seulement une partie de cet objet est visible.
- Quand une partie de l'objet dynamique est statique et une autre partie est dynamique. Dans ce cas, la partie statique de cet objet peut être considérée comme une partie du background.
- Quand un objet dynamique possède les mêmes couleurs que le background ou que l'image a peu de sources lumineuses.
- Quand la caméra est affectée par trop de mouvements comme une combinaison de tremblements et de zooms.

De ce fait, les séquences d'images suivantes sont choisies pour tester les défis cités précédemment :

- La figure 5.8 est une séquence de 6 images avec un objet dynamique pour tester l'occlusion.



FIGURE 5.8 – Série Car.

- La figure 5.9 est une séquence de 6 images avec un objet dynamique unique qui se déplace lentement.



FIGURE 5.9 – Série Cat.

- La figure 5.10 est une séquence de 6 images capturées en temps nocturne avec différentes sources de lumières et différents objets dynamiques où la caméra reste statique.



FIGURE 5.10 – Série Shibuya.

- La figure 5.11 est une séquence de 12 images avec un seul objet dynamique qui possède des couleurs similaires au background flouté avec le mouvement de cet objet qui est rapide.



FIGURE 5.11 – Série Vault.

- La figure 5.12 est une séquence de 12 images avec un seul objet dynamique et une caméra qui subit plusieurs mouvements et zooms simultanément.

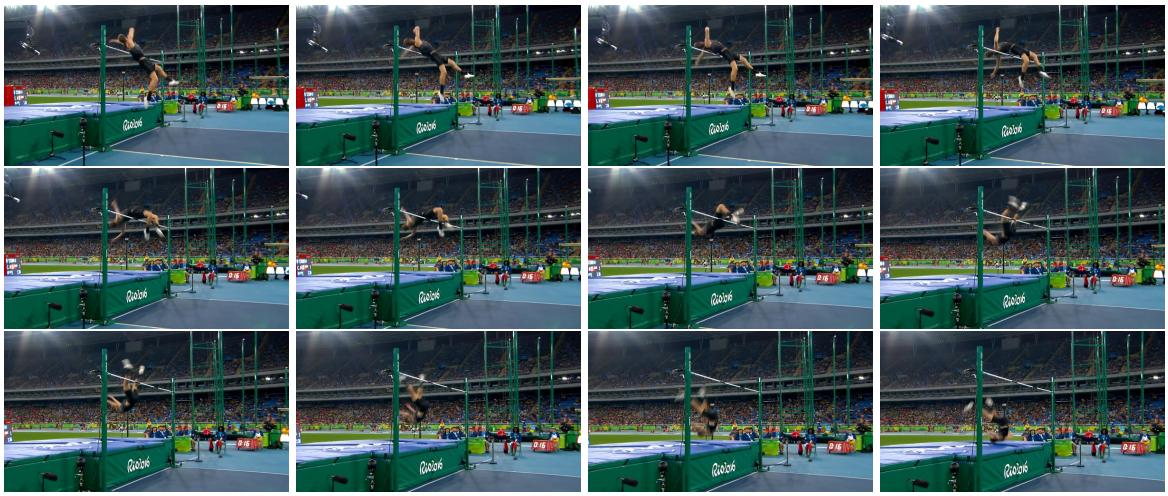


FIGURE 5.12 – Série High-jump.

5.4.1 Résultats

Les résultats de la méthode développée ont été comparés avec la méthode **RoD** d'Uyttendaele et al. [62] (Secion 2.3.2.1), la méthode **CSDP** de Zeng et al. [55] (Section 2.3.1.2), la méthode **graphcut** de Li et al. [56] (Section 2.3.1.4) et la méthode **fBm_A*** basée sur un fBm modifié et l'algorithme A* (Chapitre 4) [78]. Les résultats graphiques ont été comparés de deux manières : visuellement et avec les mesures de similarité la "Structure Similarity Index Measure" (SSIM)[65] (Section 2.4.2.1) et la "Multi-Scale Structure Similarity Index Measure (MS-SSIM)" [67] (Section 2.4.2.2).

Les images dans les figures 5.13, 5.14, 5.15, 5.16 et 5.17 montrent les résultats obtenus avec les méthodes RoD, CSDP, graphcut, fBm_A* et la méthode développée. En plus, les résultats des mesures de similarité SSIM et MS-SSIM de chaque méthode sont présentés dans le tableau 5.1.

Toutes les méthodes ont été codées en Java avec la librairie OpenCV avec un PC portable dont le microprocesseur est Intel Core i7-3632QM CPU à 2.20 GHz.

Le premier test que nous avons voulu réaliser consiste à tester les occlusions avec la série de la figure 5.8. Dans cette figure, un objet dynamique large passe derrière un objet statique. Les résultats dans la figure 5.13 montre un bon résultat pour toutes les méthodes. Cependant, dans le tableau 5.1, malgré que toutes les méthodes ont donné un bon résultat, les valeurs SSIM et MS-SSIM sont les plus élevées pour la méthode développée, ce qui peut être expliqué par le système de recommandation retournant des valeurs pour la mosaïque qui sont plus optimaux.

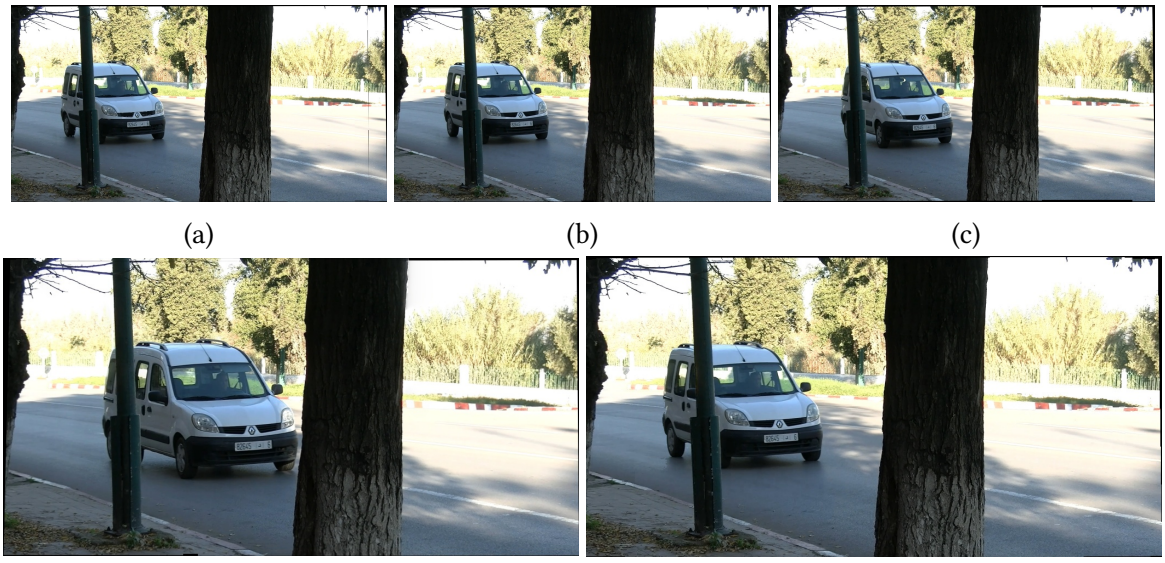


FIGURE 5.13 – Résultats obtenus de la série Car (Figure 5.8), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode.

	RoD	CSDP	Graphcut	fBm_A*	Notre méthode
SSIM					
Fig. 5.8	0, 6927	0, 7084	0, 6718	0, 7890	**0,7992**
Fig. 5.9	0, 6903	0, 5379	0, 5236	0, 7165	**0,8800**
Fig. 5.10	0, 6840	0, 5903	0, 6670	0, 7223	**0,7601**
Fig. 5.11	0, 5661	0, 6913	0, 6787	0, 7100	**0,7991**
Fig. 5.12	0, 6353	0, 6289	0, 6873	0, 8109	**0,8533**
MS-SSIM					
Fig. 5.8	0, 9331	0, 9528	0, 9141	0, 9716	**0,9723**
Fig. 5.9	0, 9135	0, 8841	0, 8759	0, 9056	**0,9436**
Fig. 5.10	0, 8840	0, 8158	0, 7679	0, 9192	**0,9534**
Fig. 5.11	0, 7571	0, 9615	0, 9542	0, 9512	**0,9815**
Fig. 5.12	0, 9145	**0,9740**	0, 9201	0, 9471	0, 9616

TABLE 5.1 – Valeurs SSIM et MS-SSIM de : RoD, CSDP, Graphcut, fBm_A* et notre méthode pour les mosaïques obtenues des figures 5.8, 5.9, 5.10, 5.11, 5.12. (Valeurs optimaux en gras)

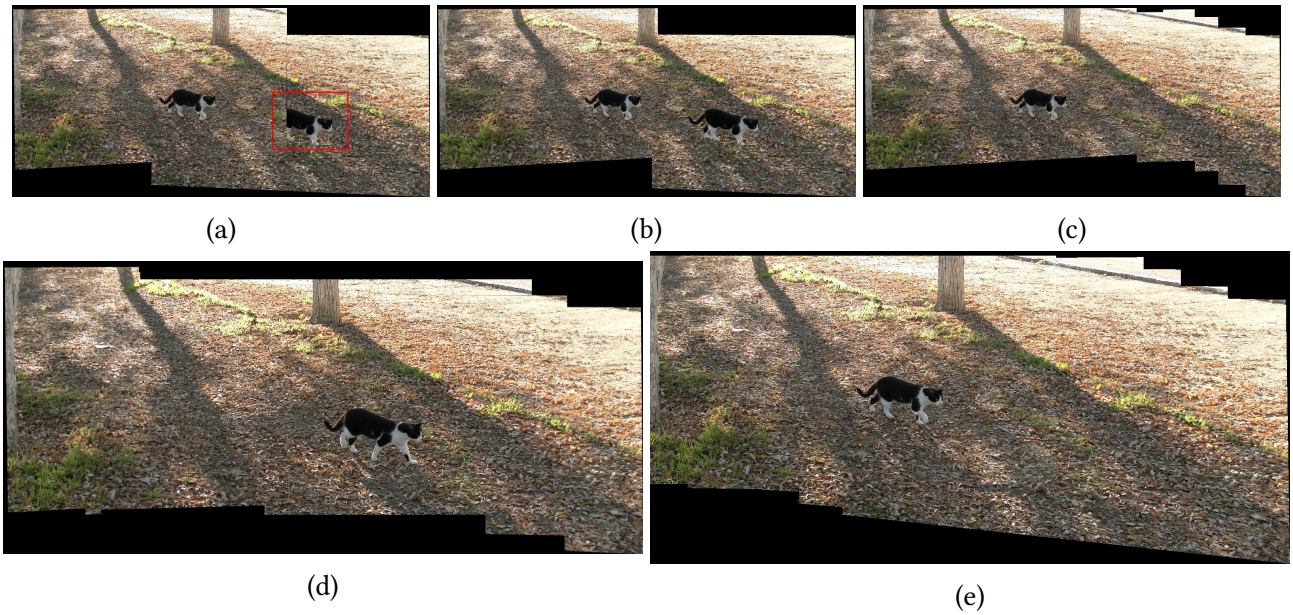
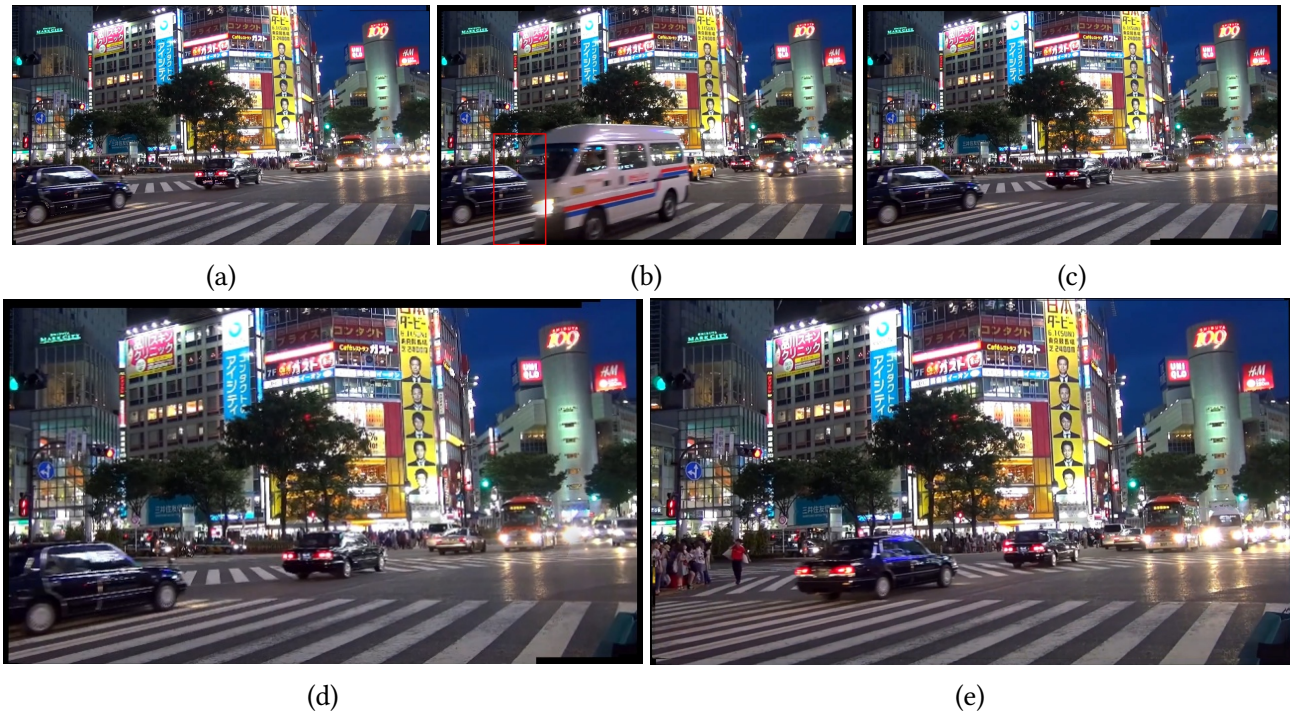


FIGURE 5.14 – Résultats obtenus de la série Cat (Figure 5.9), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode.

La seconde expérimentation est un suivi de mouvement d'un seul objet qui se déplace d'une manière lente pour que certaines parties de l'objet reste commune dans les zones de chevauchements (Figure 5.9). Les mosaïques obtenus montrent : un mauvais résultat pour la méthode RoD (Figure 5.14.a) avec du ghosting visible, des résultats moyens pour CSDP (Figure 5.14.b) où une duplication d'objets est visible et de bons résultats pour la méthode graphcut (Figure 5.14.c), la méthode fBm_A* (Figure 5.14.d) et la méthode développée (Figure 5.14.e). En effet, notre méthode a généré de bons résultats parce que les pixels similaires ont été catégorisés ensemble. Du coup, même si des parties d'un objet dynamique restent statiques dans les images d'entrées, ces parties sont quand même considérées comme des objets dynamiques. La table 5.1 confirme ces remarques puisque les valeurs SSIM et MS-SSIM sont les plus élevées pour notre méthode.

Le troisième test est effectué avec une caméra statique, plusieurs objets dynamiques et sources lumineuses (Figure 5.10). Des résultats presque optimaux ont été obtenus par CSDP (Figure 5.15.b) où un ghosting léger est visible, et de bons résultats ont été générés par Rod (Figure 5.15.a), graphcut (Figure 5.15.c), la méthode fBm_A* (Figure 5.15.d) et notre méthode (Figure 5.15.e). Ces résultats sont confirmés par les mesures SSIM et MS-SSIM optimales obtenues par notre méthode dans le tableau 5.1.

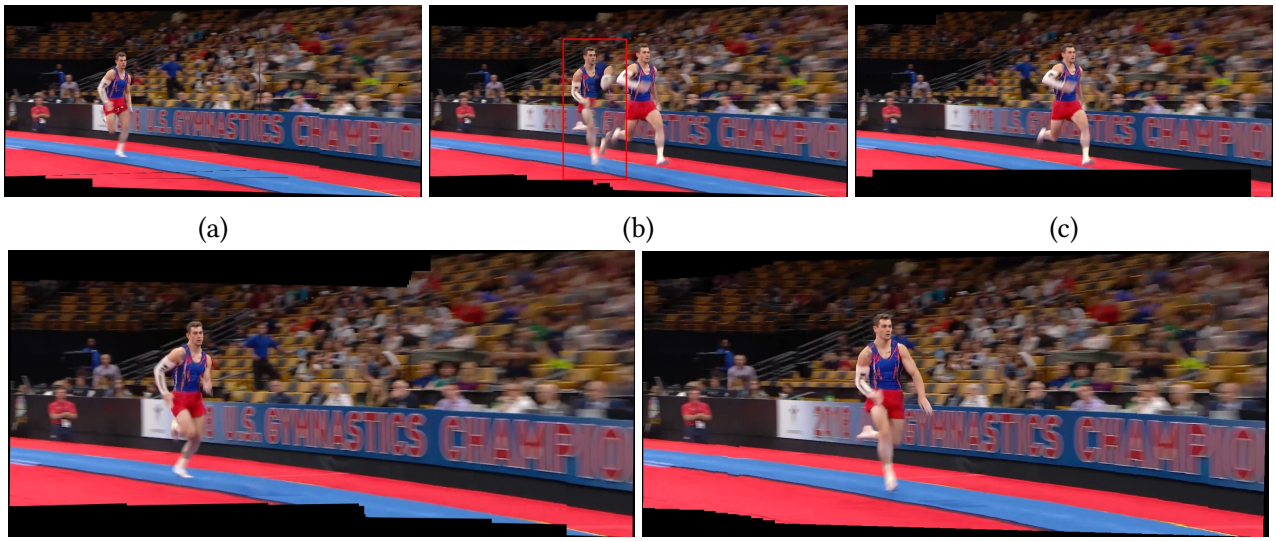


(a) (b) (c)

(d) (e)

FIGURE 5.15 – Résultats obtenus de la série Shibuya (Figure 5.10), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode.

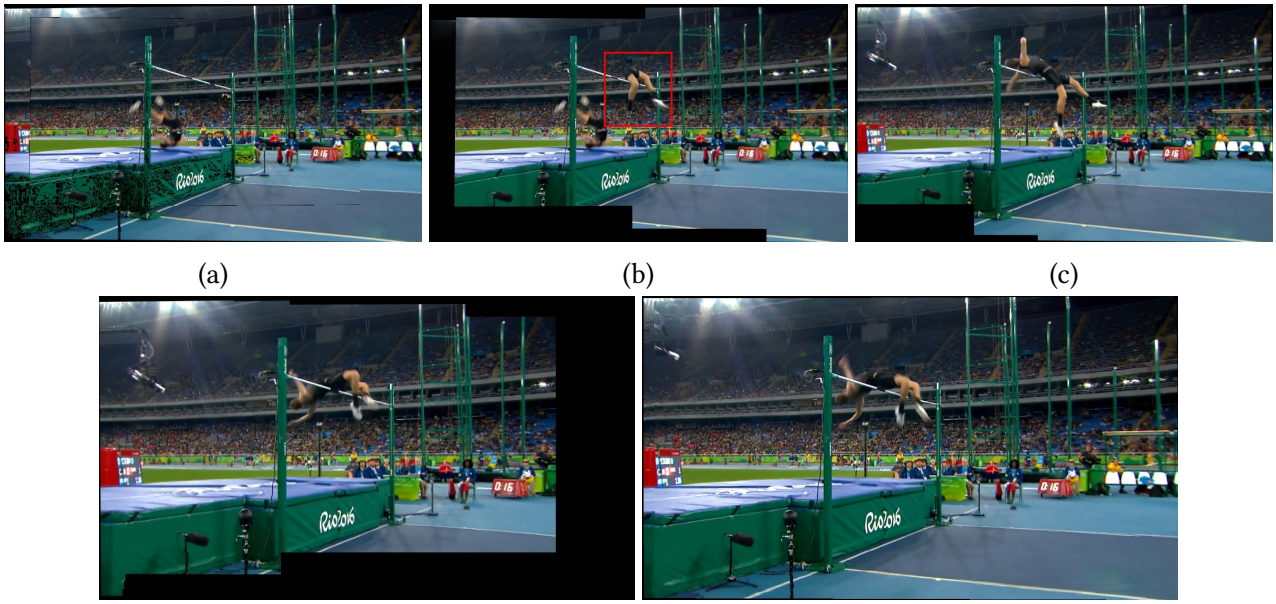
La quatrième expérimentation consiste à suivre le mouvement d'un seul objet dynamique avec des couleurs similaires que le background. En effet, quand un objet possède des couleurs similaires que le background c'est probable qu'il soit considéré comme faisant partie de ce dernier. Deux séquences d'images sont utilisées pour effectuer ce test. La première séquence d'images (Figure 5.11) utilise une caméra qui bouge avec l'objet en gardant l'objet dans le centre, puisque l'objet est très rapide, l'arrière plan est devenu flou, ce qui peut aussi jouer un rôle dans l'obtention de mauvais résultats. La seconde séquence d'images (Figure 5.12) utilise un seul objet dynamique mais avec une caméra qui subit plusieurs transformations avec un mouvement et un zoom. Pour la première séquence (Figure 5.11), de mauvais résultats ont été obtenus pour la méthode CSDP (Figure 5.16.b) et de bons résultats pour la méthode RoD (Figure 5.16.a), graphcut (Figure 5.16.c), la méthode fBm_A* (Figure 5.16.d) et la méthode développée (Figure 5.16.e). Tandis que la deuxième séquence (Figure 5.12), des résultats moyens pour RoD (Figure 5.17.a) et CSDP (Figure 5.17.b) et des bons résultats pour la méthode grahcut (Figure 5.17.c), la méthode fBm_A* (Figure 5.17.d) et la méthode développée (Figure 5.17.e). C'est résultats sont confirmés avec les mesures du tableau 5.1.



(a) (b) (c)

(d) (e)

FIGURE 5.16 – Résultats obtenus de la série Vault (Figure 5.11), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode.



(a) (b) (c)

(d) (e)

FIGURE 5.17 – Résultats obtenus de la série High-jump (Figure 5.12), (a) Rod (b) CSDP (c) Graphcut (d) fBm_A* (e) Notre méthode.

La figure 5.18 présente le temps d'exécution des différentes méthodes. Ces temps d'exécution n'incluent pas le temps de la phase du recalage d'images et incluent seulement le temps de création de la mosaïque. Ils sont aussi calculés avec une moyenne obtenue en exécutant les algorithmes 10 fois.

A partir de cette figure, nous notons que notre méthode est plus rapide que les autres méthodes. Ce résultat est expliqué par premièrement, l'utilisation de toutes les images simultanément au lieu de l'utilisation des images par paires et deuxièmement par, aucune utilisation d'algorithme de "blending" puisque le système de recommandation choisi directement le meilleur résultat possible.

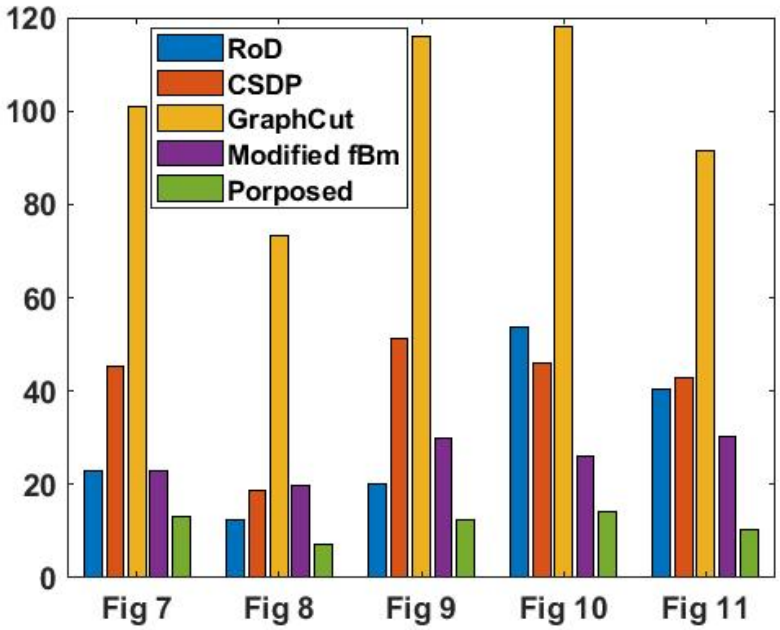


FIGURE 5.18 – Temps d'exécution des différentes approches.

5.4.2 Analyses des mesures obtenues

La méthode développée a surpassé les autres méthodes et offre plusieurs avantages. En effet, notre méthode utilise toutes les images simultanément dans un algorithme k-means pour créer des catégories. Cette opération permet d'assurer que tous les objets dynamiques sont groupés dans la même catégorie. Ainsi, il y aura qu'un seul objet dynamique qui a été recommandé d'une seule image dans la mosaïque finale puisque tous les pixels de la catégorie d'objets dynamiques auront la même recommandation. Par conséquence, les erreurs comme la duplication d'objets et le ghosting sont évités. Effectivement, à partir des mesures calculées dans le tableau 5.1, nous remarquons en moyenne :

- Pour la mesure **SSIM** : une amélioration de +25,89% par rapport à la méthode RoD d'Uyttendaele et al., de +31,29% par rapport à la méthode CSDP de Zeng et al., de +28,57% par rapport à la méthode Graphcut de Li et al. et de +9,42% par rapport à la méthode basée sur un FBm et A*.
- Pour la mesure **MS-SSIM** : une amélioration de +10,02% par rapport à la méthode RoD d'Uyttendaele et al., de +5,29% par rapport à la méthode CSDP de Zeng et al., de +9,12% par rapport à la méthode Graphcut de Li et al. et de +2,54% par rapport à la méthode basée sur un FBm et A*.

De plus, un gain de temps significatif est obtenue en utilisant toutes les images en même temps. En moyenne, nous avons réalisé un gain de temps de +54,09% par rapport à la méthode RoD d'Uyttendaele et al., de +70,01% par rapport à la méthode CSDP de Zeng et al., de +88,06% par rapport à la méthode Graphcut de Li et al. et de +55,31% par rapport à la méthode basée sur un FBm et A*.

Additionnellement, le système de recommandation utilise tous les pixels des images d'entrées. Ainsi, le résultat mosaïque de notre méthode contient tous les pixels des images d'entrées différemment des autres méthodes où certains pixels ne sont pas utilisés. Cet avantage peut être observé sur

les différentes mosaïques d'images générées par les différentes méthodes de cette section.

A partir de la table 5.1 et la figure 5.18, ainsi que les différentes mosaïques obtenues dans les figure 5.13, 5.14, 5.15, 5.16 et 5.17, nous pouvons conclure que la méthode développée génère de bons résultats dans différentes circonstances.

5.5 Conclusion

Dans ce chapitre, nous avons développé une nouvelle méthode pour la création de mosaïques d'images dynamiques en introduisant un système de recommandation. D'abord, les pixels ont été catégorisés en utilisant l'algorithme k-means, puis les profils des pixels ont été calculés. Ensuite, en considérant que le résultat est un nouvel utilisateur, les profils des pixels et les catégories ont été utilisés pour recommander le meilleur résultat possible avec le système de recommandation sans créer de discontinuités. De plus, les motivations initiales de ce chapitre était de créer une méthode qui utilise toutes les images d'entrées simultanément, de générer des mosaïques sans créer d'artefacts et d'introduire un système de recommandation pour obtenir de bons résultats avec un gain de temps. Ces motivations ont été réalisées et peuvent être constatées à travers les résultats expérimentaux obtenus.

Conclusion générale et perspectives

Dans ce mémoire, nous avons traité le sujet des mosaïques d'images créées à partir d'une scène contenant des objets dynamiques. Nous avons tout d'abord, dans le chapitre 1, défini les différents concepts, généralités et étapes de création des mosaïques d'images dynamiques, ensuite dans le chapitre 2, nous avons exploré les différentes méthodes de traitement des objets dynamiques de la littérature, ainsi que la procédure d'évaluation des mosaïques et les mesures associées à cette procédure.

En étudiant les différentes méthodes dans ce domaine de recherche, nous avons développé trois nouvelles méthodes de traitement des objets dynamiques pour la création d'une mosaïque dynamique.

Au début (Chapitre 3), nous avons commencé notre recherche dans la catégorie des méthodes basées sur le calcul d'une "seamline". L'idée initiale était d'essayer de diviser les différentes parties des images en des zones où aucun artefact ne peut être créé et des zones où c'est probable d'obtenir des erreurs dans la mosaïque d'images finale. Nous avons réussi à réaliser cette opération en utilisant les "inliers" et "outliers" déjà calculés lors de la phase du recalage d'images. La définition de base des "inliers" c'est que ce sont des paires de points similaires et les "outliers" ce sont des paires de points non similaires. En profitant de ces points, nous avons utilisé le détecteur de contours Canny pour séparer l'image en plusieurs régions et classer ces régions en des régions "inliers" ou "outliers" selon le type de point qui existe dans la région. En combinant ces régions avec une différence d'intensité, la carte d'énergie qu'on a créé a permis de surligner les zones qui peuvent générer des artefacts et d'assombrir les zones sûres. Finalement, on a cherché la meilleure "seamline", qui passe par un maximum de zones "inliers" et un minimum de zones "outliers", dynamiquement parmi plusieurs "seamlines".

Malgré les bons résultats de la première méthode, cette dernière possédait néanmoins certaines limitations. Effectivement, la méthode basée sur le détecteur de Canny dépend du recalage d'images. Ainsi, dans le cas où la zone de chevauchement est petite, la méthode a plus de chance d'échouer. Et le faite que la méthode calcule plusieurs "seamlines" pour en choisir la meilleure peut coûter en temps d'exécution si les images d'entrées sont grandes. Dans l'objectif de remédier à ces problèmes, dans le chapitre 4, la seconde méthode que nous avons proposé avait pour but de calculer une "seamline" unique en utilisant l'algorithme d'apprentissage automatique A* qui grâce à son heuristique améliore considérablement le temps d'exécution. Cependant, le problème que nous avons rencontré c'est celui des zones similaires dans les zones de chevauchements qui créent des zones homogènes dans la carte d'énergie. Ainsi, avec des pixels similaires, le temps de calcul de A* devient très élevé et

n'est plus adapté pour le calcul d'une "seamline". Pour cette raison, nous avons cherché à créer une carte d'énergie où chaque pixel possède une valeur unique et en même temps, il reflète les valeurs de son voisinage. La solution que nous avons proposé c'est d'utiliser un mouvement fractionnaire Brownien (fBm) avec une fonction de similarité "Zero Normalized Cross Correlation" (ZNCC) au lieu d'une fonction de bruit aléatoire. La difficulté pour créer cette carte d'énergie était d'attribuer les différentes propriétés du fBm au fBm modifié notamment l'effet d'auto-similarité. Néanmoins, nous avons réussi à créer cette carte d'énergie avec les paramètres de la fréquence, nombre de distributions et persistance avec plusieurs analyses qui ont été effectuées sur ces paramètres pour en choisir les valeurs optimales.

Après plusieurs tests effectués sur la première et seconde méthode avec les différentes autres méthodes dans le domaine, nous avons remarqué que la majorité des méthodes utilisent deux ou trois images au maximum pour traiter les objets dynamiques. Afin de régler ce problème, nous avons créé une méthode dans le chapitre 5 qui permet d'utiliser toutes les images simultanément pour générer la mosaïque finale. Nous avons opté pour l'utilisation d'un système de recommandation. L'idée principale était de considérer les images comme des utilisateurs, les pixels des images comme des objets et les valeurs obtenues des pixels comme le score attribué à l'objet, par l'utilisateur. Ainsi, la mosaïque va être considérée comme un nouvel utilisateur qui possède des pixels (objets) qui n'ont pas de valeurs, et avec les différents scores de chaque image, le système de recommandation va remplir chaque pixel de la mosaïque en recommandant la meilleure valeur parmi les pixels disponibles. En outre, vu que le système de recommandation va être basé sur les objets, nous avons opté pour un système de recommandation "item-to-item" avec un filtrage collaboratif. Afin de pouvoir utiliser les pixels, il fallait créer des profils pour ces pixels. Au début, les attributs des profils ont été calculés avec les valeurs de chaque pixel, ce qui dans le cas parfait devrait être suffisant. Nonobstant, vu que des erreurs de projection existeront toujours et ces erreurs s'accumulent quand le nombre d'images est élevé, le résultat obtenu était flou. De plus, vu que la recommandation était effectuée par pixel, la mosaïque résultante supprimait tous les objets dynamiques et créait une sorte de reconstruction de scène. Afin de régler ce problème, nous avons introduit la catégorie des pixels en regroupant tous les pixels en des classes avec l'algorithme de "clustering" k-means. En ajoutant ces catégories, tous les pixels d'une catégorie ont été recommandés en même temps, ce qui a enlevé l'effet flou et la suppression d'information pour obtenir une mosaïque de haute qualité.

Après une étude détaillée sur les mosaïques d'images spécifiques aux scènes contenant des objets dynamiques, nous avons constaté que c'est un domaine qui rencontre beaucoup de difficultés, ce qui en fait un domaine qui possède un potentiel pour se développer dans le futur.

La difficulté principale c'est la comparaison des méthodes. En effet, la comparaison actuelle dans le domaine des mosaïques d'images dynamiques n'est pas optimale pour plusieurs raisons, ce qui fait que ce sujet d'étude peut être développé. Premièrement, absence d'une base de données pour tester les méthodes, même la base de données DAVIS utilisée dans les différents tests dans ce projet n'est pas

adapté aux mosaïques d'images dynamiques mais plutôt pour la segmentation d'objets dynamiques et leur suivis (des bases de données existent pour les mosaïques statiques). Deuxièmement, la comparaison d'une méthode de mosaïque d'images dynamiques avec une autre méthode du même type est effectuée en comparant chaque méthode avec les images d'entrées et non pas entre les méthodes, cette comparaison est dû au faite que chaque méthode génère une mosaïque unique. Cependant, cette manière de comparer peut considérer des résultats contenant des erreurs, comme la duplication d'objets, optimaux.

Suite à l'utilité du domaine des mosaïques et le faite qu'il est souvent utilisé comme un outil dans plusieurs domaines d'infographie et de vision par ordinateur, le domaine des mosaïques d'images est un sujet qui demandera encore plus de recherches. En effet, il peut être revalorisé vers un système qui fonctionne en temps réel où la principale difficulté va être de gérer les objets dynamiques en temps réel surtout pour éviter la duplication d'objets.

Une des prochaines étapes pour les mosaïques est les vidéos mosaïques, cependant au lieu d'utiliser une vidéo pour créer une seule image. Eventuellement, une recherche peut être effectuée pour utiliser plusieurs vidéos pour créer une seule vidéo. La principale difficulté si c'est dans le cas des scènes dynamiques va être de gérer les objets dynamiques dans l'axe du temps. En effet, avec le facteur du temps, un objet dynamique spécifique doit être présent dans tous les frames pour ne pas obtenir un effet de clignotement dans la vidéo et s'il change de zones il ne doit pas soudainement disparaître.

Plusieurs recherches éventuelles peuvent être effectuées sur les méthodes que nous avons créé, notamment pour le système de recommandation pour régler le problème du "*cold start*" qui apparaît quand deux images sont utilisées. Dans le cas de deux images, combiner ce système avec une méthode basée sur une "seamline" peut régler le problème.

Annexe **A**

Outils mathématiques

A.1 Introduction 119
A.2 “Least square method” 119
A.3 Décomposition en valeurs singulières (SVD) 120
A.4 Différentes interpolations 121
A.5 Conclusion 122

A.1 Introduction

Durant les différentes recherches sur la création des mosaïques, plusieurs méthodes mathématiques qu'on retrouve souvent sont utilisées. Dans cette annexe, nous allons présenter les plus utilisées.

A.2 "Least square method"

La méthode des moindres carrés ("Least square method" (LSM)) est une procédure statistique qui permet de trouver une solution approximative unique à un ensemble de points. En mosaïques d'images, on l'utilise pour résoudre un système d'équation linéaire où la solution est appelée la *best fit line*, le type d'équation à résoudre est de la forme de :

$$\underbrace{\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}}_{\mathrm{b}} \tag{A.1}$$

Où :

- x est un vecteur de taille n qui contient les variables inconnues avec le nombre d'inconnus qu'on souhaite chercher est n.
- A est une matrice de taille (m, n) contenant les coefficients de chaque équation avec le nombre d'équations est m.
- b un vecteur de taille m qui contient la solution de chaque équation.

La méthode des moindres carrés est utilisée dans les mosaïques d'images dans la partie du prétraitement pour calculer une correction d'exposure. La solution de l'équation A.1 est calculée de la façon suivante :

$$x = (A^T A)^{-1} A^T b \tag{A.2}$$

Tel que :

- A^T est la transposée de A.
- $(A^T A)$ est inversible.

A.3 Décomposition en valeurs singulières (SVD)

La décomposition en valeurs singulières permet de décomposer une matrice A de taille (m, n) en trois matrices (Equation A.3).

$$A = U \times \sum \times V^T \tag{A.3}$$

Où :

- U une matrice unitaire orthogonale de taille (m, m).
- V une matrice unitaire orthogonale de taille (n, n).
- $\sum$ une matrice diagonale avec des coefficients réels positifs de taille m, n

Dans les mosaïques d'images, on rencontrera souvent sur une équation de type A.4 qui est une équation linéaire des moindres carrés homogène ("Homogeneous Linear Least Squares") comparé à la version non homogène de l'équation A.1.

$$\underbrace{\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}}_{\text{A}} \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_{\text{x}} = 0 \tag{A.4}$$

Avec x un vecteur de taille n contenant le nombre d'inconnus qu'on souhaite chercher.

Dans le cas où l'équation est homogène, la solution présentée précédemment (A.2) ne fonctionne plus et c'est une décomposition en valeurs singulières qui est effectuée. La solution de cette équation va être le vecteur singulier le plus à droite de la matrice V qui correspond à la valeur singulière la plus petite σ_n.

Les étapes pour effectuer une décomposition en valeurs singulières sont les suivantes :

- Calcul des valeurs singulières σ_i avec le produit $(A^T A)$ où A^T est la transposée de A. Le nombre des valeurs singulières est déterminé par le rang de la matrice A, ainsi si $n < m$ alors $rang(A) = n$ sinon $rang(A) = m$.
- Extraction des vecteurs propres pour calculer U et V. En effet, l'extraction de m vecteurs propres $(u_1, u_2, \dots, u_m)$ de U est effectuée sur la matrice (AA^T) de taille (m, m) et l'extraction de n vecteurs propres $(v_1, v_2, \dots, v_n)$ de V est effectuée sur la matrice $(A^T A)$ de taille (n, n).
- Calcul des signes s_i pour chaque valeur avec l'équation $As_i v_i = \sigma_i u_i$ pour que les valeurs des vecteurs propres de la matrice (Soit U Soit V) n'ayant pas le même rang que A restent positives et appliquer les signes sur la matrice qui a le même rang que A.

- Remplir les matrices avec l'équation A.5.

$$\underbrace{\begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} u_1 & u_2 \dots & u_m \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_{rang(A)} \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} v_1 & v_2 \dots & v_n \end{bmatrix}}_{V} \tag{A.5}$$

Avec :

- u_i des vecteurs propres de taille $(m, 1)$.
- v_i des vecteurs propres de taille $(n, 1)$.

Les valeurs de la dernière colonne de V (le vecteur propre v_n) sont la solution de l'équation A.4.

A.4 Différentes interpolations

Dans le domaine du traitement images, les pixels ont des coordonnées entiers. Cependant, quand les pixels subissent plusieurs transformations, notamment dans leurs positions, c'est possible d'obtenir des coordonnées réelles. Si ces coordonnées réelles sont utilisées pour afficher l'image, les coordonnées sont converties en entier. Par conséquent, certaines valeurs sont écrasées ce qui peut baisser la qualité de la mosaïque. C'est pourquoi différentes interpolations peuvent être utilisées pour régler ce problème.

Parmi les interpolations il y a l'interpolation nearest, linéaire, cosinusoïdale et cubique.

L'interpolation la plus simple est l'interpolation *nearest* qui administre la valeur du coordonnée réelle le plus proche du pixel de coordonnées entier (x, y).

L'interpolation linéaire utilise les deux points les plus proches et calcule la valeur de ces deux points avec l'équation A.6 en supposant que $(x_1, y_1) < (x, y) < (x_2, y_2)$.

$$f_{linaire}(x, y, x_1, y_1, x_2, y_2) = f(x_1, y_1) \times (1 - d) + f(x_2, y_2) \times d \tag{A.6}$$

Avec :

- $f_{linaire}(x, y, x_1, y_1, x_2, y_2)$ la valeur du point avec les coordonnées entiers qu'on souhaite calculer à partir des deux points réels (x_1, y_1) et (x_2, y_2).
- $f(x_1, y_1)$ et $f(x_2, y_2)$ les valeurs des points avec les coordonnées réelles (x_1, y_1) et (x_2, y_2) respectivement.
- d une valeur appartenant entre 0 et 1 calculée avec une distance tel quel : $d = \frac{distance((x,y),(x_1,y_1))}{distance((x_2,y_2),(x_1,y_1))}$.

L'interpolation cosinusoïdale utilise deux points et les mêmes variables que l'interpolation linéaire, cependant elle est obtenue avec l'équation A.7 où une nouvelle variable est calculée avec un cosinus. L'interpolation sinusoïdale est calculée de la même façon sauf un sinus est utilisé au lieu d'un cosinus.

$$f_{cosinusoidale}(x, y, x_1, y_1, x_2, y_2) = f(x_1, y_1) \times (1 - v) + f(x_2, y_2) \times v \tag{A.7}$$

Avec : $v = (1 - cos(d \times \pi))/2$

Finalement, l'interpolation cubique est calculée à partir de quatre points (x_1, y_1), (x_2, y_2), (x_3, y_3) et (x_4, y_4). Il existe plusieurs moyens de la calculer, mais le plus simple est d'utiliser l'interpolation cosinusoïdale. On suppose que les quatre points forment un carré autour du point, les étapes sont les suivantes :

- Premièrement, l'interpolation cosinusoïdale est utilisée entre les points (x_1, y_1) et (x_2, y_2) pour calculer la valeur du point $i_1 = (x_1^{'}, y)$ qui a la même valeur de l'axe des ordonnées y que le point (x, y).
- Deuxièmement, la même opération est effectuée pour calculer la valeur $i_2 = (x_2^{'}, y)$ ayant la même valeur y que le point (x, y).
- Finalement, la valeur du pixel (x, y) est calculée avec une interpolation entre les points i_1 et i_2.

A.5 Conclusion

Dans cet annexe, nous avons présenté les différents outils mathématiques utilisés dans cette thèse.

Annexe **B**

Utilisation du flux optique dense pour créer une mesure quantitative pour l'évaluation d'une mosaïque

B.1 Introduction 124
B.2 Flux optique dense 124
B.3 Résultats expérimentaux 125
B.4 Conclusion 128

B.1 Introduction

Dans cette annexe, nous allons présenter une nouvelle méthode [70] pour calculer une mesure quantitative qui permet d'évaluer les mosaïques d'images dynamiques en utilisant un flux optique dense [86] combiné avec la "Structure Similarity Index Measure" (SSIM) [65] (Section 2.4.2.1).

Dans la section 2.4.1 nous avons expliqué la procédure d'évaluation des mosaïques. En effet, la mosaïque est évaluée en utilisant des mesures qui sont calculées sur des blocs pour chaque pixel entre plusieurs images et la valeur maximale est choisie comme mesure pour ce pixel précis. Cependant, l'utilisation de blocs d'une taille prédéfinie est naïve et peut engendrer de fausses évaluations. Ainsi si le bloc est petit, et si la totalité des pixels du bloc font partie d'un artefact, la mesure obtenue va quand même être optimale vu que par définition la valeur maximale est choisie. Alors que si le bloc est trop grand et que l'artefact dans le bloc est petit, la mesure peut être optimale vu que les pixels qui créent l'artefact n'auront pas un impact sur la moyenne.

C'est ainsi qu'au lieu d'utiliser des blocs avec des tailles prédéfinies, nous allons chercher les champs de déplacements des pixels et créer des zones où les mesures peuvent être calculées.

Tout d'abord, nous allons utiliser un flux optique dense dans la zone de chevauchement des images d'entrées après leur projection dans le même système de coordonnées afin d'estimer le mouvement des pixels. Par conséquent, c'est possible de séparer les pixels qui ont les mêmes types de mouvements. Ensuite, nous allons utiliser les zones trouvées avec le flux optique dense pour calculer la valeur de SSIM dans chaque zone. Finalement, la valeur de SSIM finale va être la moyenne des valeurs sur les zones pouvant créer des artefacts.

Le recalage d'image a été effectué avec la méthode SIFT [24] (Section 1.3.4.1) et la mise en correspondance avec RANSAC [38] (Section 1.3.4.2).

B.2 Flux optique dense

Le flux optique dense [86] est une méthode de flux optique. Cette méthode estime les mouvements, ou le champ de déplacement, à partir de deux images d'entrées en essayant d'équilibrer les mouvements de l'arrière-plan. De plus, le flux optique dense montre les mouvements spatiaux et n'inclue pas le facteur du temps.

A la différence aux méthodes du flux optique clairsemé qui utilise un certain nombre de pixels, le flux optique dense utilise tous les pixels des images. En effet, cette méthode utilise des transformées polynomiales exactes pour calculer le champ de déplacement et retourne le vecteur de déplacement $(\vec{u}, \vec{v})$ pour chaque pixel.

Pour pouvoir visualiser la position des déplacements et pour obtenir les zones qui vont être utilisées pour la comparaison, la magnitude et l'orientation entre $\vec{u}$ et $\vec{v}$ sont calculées :

- La magnitude G amplifie les valeurs. D'une part, elle montre les déplacements qui sont restés constants si la magnitude est petite, et de l'autre part, elle montre les pixels qui ont fait des grands mouvements si la magnitude est élevée. De plus la valeur est normalisée pour que $G(x, y) \in [0, 1]$. L'équation B.1 montre la formule.

$$G(x, y) = \sqrt{\vec{u}(x, y)^2 + \vec{v}(x, y)^2} \tag{B.1}$$

  Où x et y sont les coordonnées d'un pixel (x, y).

- L'orientation θ montre la direction principale des changements de déplacements. Elle est calculée par l'équation B.2. Normalement, la valeur obtenue est entre $[0°, 360°]$ mais la valeur est normalisée également pour que $\theta(x, y) \in [0, 1]$.

$$\theta(x, y) = arctan\frac{\vec{v}(x, y)}{\vec{u}(x, y)} \tag{B.2}$$

  Où x et y sont les coordonnées d'un pixel (x, y).

Après avoir obtenu les magnitudes et orientations normalisées entre les vecteurs $\vec{u}$ et $\vec{v}$) de chaque pixel, une image dans l'espace couleur HSV (hue, saturation, value) est créée en attribuant l'angle au canal hue H, la magnitude au canal value V et en initialisant la valeur du canal de saturation S à 1 pour chaque pixel.

Afin d'obtenir une meilleur sélection des zones dynamiques, la magnitude normalisée est divisée en dix parties où chaque partie est arrondie à la valeur la plus proche par l'équation B.3. Cette opération va seuiller le flux optique dense et rendra les zones détectées plus précises où chaque zone non connectée va être utilisée dans la comparaison.

$$G(x, y) = round(G(x, y) * 10) * 0.1 \tag{B.3}$$

En transformant l'image qui est dans l'espace couleur HSV à l'espace couleur RGB, les zones de déplacements deviennent visibles. La figure B.1 montre la zone de chevauchement de deux images (Figure B.1.a), le flux optique dense obtenu (Figure B.1.b) à partir de ces deux images, ainsi que les différentes zones obtenues grâce au seuillage (Figure B.1.c)

B.3 Résultats expérimentaux

Dans cette section, nous allons comparer les mesures obtenues avec la valeur de SSIM obtenue en utilisant les zones créées avec le flux d'optique dense (SSIM_DOF) et les valeurs de SSIM avec des blocs de 16 et de 256.

Les images utilisées pour la comparaison sont prises de la base de données DAVIS [77]. La figure B.2 montre les zones de chevauchements des images d'entrées (Après le recalage d'image) des 5 séries



(a)



(b) (c)

FIGURE B.1 – Exemple de création de zones avec le flux optique dense, (a) Images d'entrées, (b) Flux optique dense des images, (c) Extractions de zones.

et les zones de chevauchements de 3 méthodes : une méthode statique qui effectue un simple alpha blending (Section 1.4.1.1), une méthode qui calcule une "seamline" et crée des artefact [54] (Section 2.3.1.1) et une méthode qui calcule une "seamline" avec de bons résultats (Chapitre 3).

La mesure SSIM_DOF développée, la mesure SSIM et les méthodes utilisées dans la figure B.2 ont été codées en Java en utilisant la librairie OpenCV avec un PC portable ayant un microprocesseur Intel Core i7-3632QM CPU à 2.20 GHz.

Le tableau B.1 montre les résultats des mesures quantitatives des mosaïques créées avec différentes approches, avec SSIM calculé avec un bloc de 16 et un bloc de 256 et le SSIM calculé avec les zones générées du flux d'optique dense, obtenues à partir des zones de transitions de la figure B.2.

La première méthode crée une mosaïque avec l'alpha blending sans traitement d'objets dynamiques. Puisque l'approche ne gère pas les artefacts, nous remarquons que les résultats sont d'une mauvaise qualité ce qui peut également être observé dans le tableau B.1 où toutes les valeurs ne sont pas optimales. La mesure développée donne des valeurs moins optimales que les autres mesures de SSIM ce qui reflète les résultats mosaïques.

La seconde méthode est une méthode basée sur le calcul d'une "seamline" qui a créé des artefacts. En effet, les artefacts créés dans certaines images sont visibles comme l'objet de la figure B.2.a.3 qui a

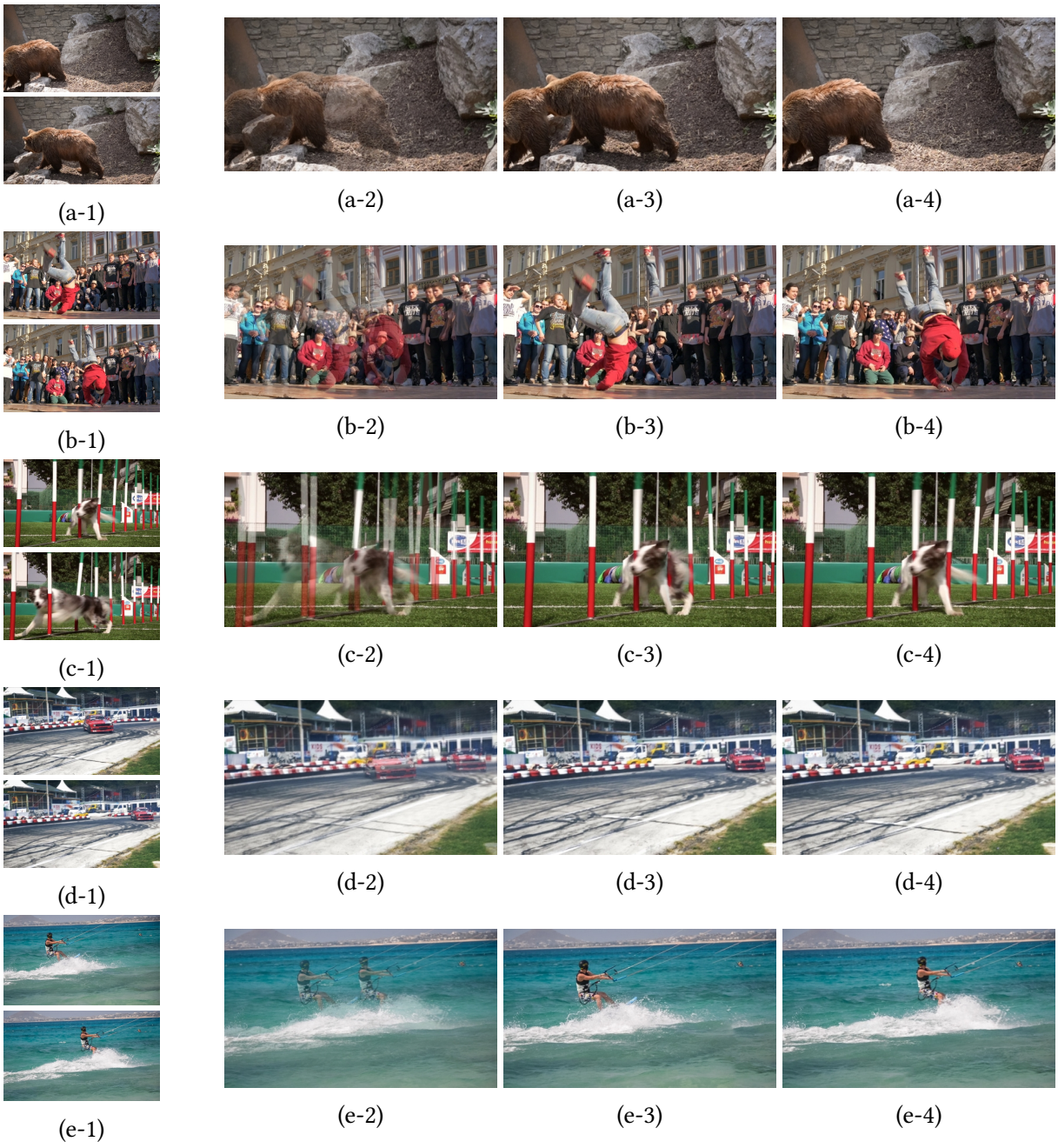


FIGURE B.2 – Images utilisées pour la comparaison des mesures tirées de la base de données DAVIS avec (a) bear, (b) breakdance, (c) dog-agility, (d) drift-straight, (e) kite-surf et (1) la zone de chevauchement des images, (2) résultat d'une méthode statique (3) mauvais résultat d'une méthode basée sur une "seamline" (4) bon résultat d'une méthode basée une "seamline".

été coupé, un objet (jambe) dans la figure B.2.b.3 qui est visible, ou un poteau disparu dans la figure B.2.c.3. Tandis que dans certaines images les artefacts ne sont pas visibles comme dans la figure B.2.d.3 où il y a beaucoup d'effets parallaxes où certaines lignes dans la route sont non continues ou dans le cas de la figure B.2.e.3 où la texture de l'arrière-plan n'est plus continue. Néanmoins, malgré la présence de ces erreurs, la mesure SSIM a quand même retourné des valeurs optimales. Cependant, avec l'utilisation des zones créées avec le flux optique dense, les mesures obtenues offre une meilleure représentation des résultats visuels générés.

		Figure B.2.a	Figure B.2.b	Figure B.2.c	Figure B.2.d	Figure B.2.e
Première méthode Figure B.2._.2	SSIM_16	0.9269	0.9424	0.9283	0.9251	0.9254
	SSIM_256	0.9445	0.9578	0.9343	0.9638	0.9491
	SSIM_DOF	0.8658	0.9153	0.9058	0.9105	0.9045
Deuxième méthode Figure B.2._.3	SSIM_16	0.9983	0.9996	0.9954	0.9985	0.9987
	SSIM_256	0.9803	0.9946	0.9815	0.9889	0.9952
	SSIM_DOF	0.8851	0.9586	0.9004	0.9567	0.8114
Troisième méthode Figure B.2._.4	SSIM_16	0.9983	0.9994	0.9976	0.9985	0.9984
	SSIM_256	0.9887	0.9970	0.9988	0.9928	0.9871
	SSIM_DOF	0.9998	0.9993	0.9893	0.9498	0.9605

TABLE B.1 – Valeurs des mesures d'évaluation obtenues avec les images de la figure B.2 avec SSIM avec un bloc de 16 et de 256 et la mesure de SSIM développée avec le flux optique dense.

La dernière méthode utilisée pour la comparaison est une méthode basée sur le calcul d'une "seam-line" avec de bons résultats. Dans ce cas, on remarque dans le tableau B.1 que toutes les mesures sont optimales.

B.4 Conclusion

En créant des méthodes qui génèrent des mosaïques avec des images d'entrées qui possèdent des objets dynamiques, il est difficile d'évaluer la mosaïque générée, principalement à cause du faite que chaque méthode synthétise une mosaïque unique. C'est ainsi dans cette annexe, nous avons présenté une mesure quantitative qui prend en compte les différents artefacts qui peuvent apparaître. En effet, nous avons combiné la valeur SSIM avec des régions générées par les vecteurs de déplacements du flux optique dense pour mesurer que les pixels qui peuvent créer des artefacts.

Bibliographie

[1] A. Pandey and U. C. Pati, "Image mosaicing : A deeper insight," Image and Vision Computing, vol. 89, pp. 236–257, Sep. 2019.

[2] D. T. Kim, C.-H. Cheng, and D.-G. Liu, "A Stable Video Stitching Technique for Minimally Invasive Surgery," Proceedings of the 2019 9th International Conference on Biomedical Engineering and Technology - ICBET' 19, 2019.

[3] Y. Angel, D. Turner, S. Parkes, Y. Malbeteau, A. Lucieer, and M. F. McCabe, "Automated Georectification and Mosaicking of UAV-Based Hyperspectral Imagery from Push-Broom Sensors," Remote Sensing, vol. 12, no. 1, p. 34, Dec. 2019.

[4] L. Li, J. Tu, Y. Gong, J. Yao, and J. Li, "Seamline network generation based on foreground segmentation for orthoimage mosaicking," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 148, pp. 41–53, Feb. 2019.

[5] S. Laaroussi, A. Baataoui, A. Halli, and S. Khalid, "A dynamic mosaicking method based on histogram equalization for an improved seamline," Procedia Computer Science, vol. 127, pp. 344–352, 2018.

[6] Z. Qu, T. Wang, S. An, and L. Liu, "Image seamless stitching and straightening based on the image block," IET Image Processing, vol. 12, no. 8, pp. 1361–1369, Aug. 2018.

[7] N. Gracias and S. Negahdaripour, "Underwater Mosaic Creation using Video sequences from Different Altitudes," Proceedings of OCEANS 2005 MTS/IEEE.

[8] D. Scaramuzza, "Omnidirectional Camera," Computer Vision, pp. 552–560, 2014.

[9] M. Brown and D. G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features," International Journal of Computer Vision, vol. 74, no. 1, pp. 59–73, Dec. 2006.

[10] W. Jackson, "Digital Image Compositing Fundamentals," Springer link, DOI :https ://doi.org/10.1007/978-1-4842-4060-1 , 2015.

[11] R. S. Shapiro, "Panoramic Ultrasound of the Thyroid," Thyroid, vol. 13, no. 2, pp. 177–181, Feb. 2003.

[12] J. M. Scott, D. S. Martin, R. Ploutz-Snyder, T. Matz, T. Caine, M. Downs, K. Hackney, R. Buxton, J. W. Ryder, and L. Ploutz-Snyder, "Panoramic ultrasound : a novel and valid tool for monitoring

change in muscle mass," Journal of Cachexia, Sarcopenia and Muscle, vol. 8, no. 3, pp. 475–481, Jan. 2017.

[13] L. Zhang, M. Ye, P. Giataganas, M. Hughes, and G.-Z. Yang, "Autonomous scanning for endomicroscopic mosaicing and 3D fusion," IEEE International Conference on Robotics and Automation (ICRA), May 2017.

[14] J. Chen, Q. Xu, L. Luo, Y. Wang, and S. Wang, "A Robust Method for Automatic Panoramic UAV Image Mosaic," Sensors, vol. 19, no. 8, p. 1898, Apr. 2019.

[15] I. Abaspur Kazerouni, G. Dooly, and D. Toal, "Underwater Image Enhancement and Mosaicking System Based on A-KAZE Feature Matching," Journal of Marine Science and Engineering, vol. 8, no. 6, p. 449, Jun. 2020.

[16] A. Pereira Nunes, A. R. Silva Gaspar, A. M. Pinto, and A. Castilho Matos, "A mosaicking technique for object identification in underwater environments," Sensor Review, vol. 39, no. 3, pp. 387–396, May 2019.

[17] A. Bartoli, N. Dalal, and R. Horaud, "Motion Panoramas," Computer Animation and Virtual Worlds, vol. 15, no. 5, pp. 501–517, Dec. 2004.

[18] National Aeronautics and Space Administration (NASA), (2020) Dernier accès le 01/11/2020 <https ://worldview.earthdata.nasa.gov/>

[19] P. Azzari, L. Di Stefano, F. Tombari, and S. Mattoccia, "Markerless Augmented Reality Using Image Mosaics," Image and Signal Processing, pp. 413–420, 2008.

[20] S. Jeon and G. J. Kim, "Mosaicing a Wide Geometric Field of View for Effective Interaction in Augmented Reality," 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nov. 2007.

[21] R. Caywood, J. Edwards, and R. S. Aygun, "Object mosaicking : Reconstruction of moving objects captured through a limited view," IEEE SOUTHEASTCON, Mar. 2014.

[22] H. C. Brown and J. Z. Snyder, "Optical Mosaicking and 3D Reconstruction aboard an Underway Iver3 UUV," OCEANS MTS/IEEE Charleston, Oct. 2018.

[23] D. Capel, "Registration : Geometric and Photometric," Image Mosaicing and Super-resolution, pp. 17–45, 2004.

[24] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[25] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," Computer Vision and Image Understanding, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[26] Y. Angel, D. Turner, S. Parkes, Y. Malbeteau, A. Lucieer, and M. F. McCabe, "Automated Georectification and Mosaicking of UAV-Based Hyperspectral Imagery from Push-Broom Sensors," Remote Sensing, vol. 12, no. 1, p. 34, Dec. 2019.

[27] X. Li, Z. Li, R. Feng, S. Luo, C. Zhang, M. Jiang, and H. Shen, "Generating High-Quality and High-Resolution Seamless Satellite Imagery for Large-Scale Urban Regions," Remote Sensing, vol. 12, no. 1, p. 81, Dec. 2019.

[28] J. Zhang, X. Yin, J. Luan, and T. Liu, "An improved vehicle panoramic image generation algorithm," Multimedia Tools and Applications, vol. 78, no. 19, pp. 27663–27682, Jun. 2019.

[29] P. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," Procedings of the British Machine Vision Conference, 2013.

[30] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," Procedings of the Alvey Vision Conference 1988, 1988.

[31] Z. Jiang and M. Liu, "Fast image mosaic algorithm based on the improved Harris-SIFT algorithm," International Symposium on Photonics and Optoelectronics 2015, Aug. 2015.

[32] S. K. Sharma and K. Jain, "Image Stitching using AKAZE Features," Journal of the Indian Society of Remote Sensing, vol. 48, no. 10, pp. 1389–1401, Sep. 2020.

[33] S. A. K. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Mar. 2018.

[34] J. Wu, Z. Cui, V. S. Sheng, P. Zhao, D. Su, and S. Gong, "A Comparative Study of SIFT and its Variants," Measurement Science Review, vol. 13, no. 3, pp. 122–131, Jun. 2013.

[35] V. P. Patil and U. B. Gohatre, "Impact of selecting image feature detection method for development of panorama under different light conditions," IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Sep. 2017.

[36] A. Laraqui, A. Baataoui, A. Saaidi, A. Jarrar, M. Masrar, and K. Satori, "Image mosaicing using voronoi diagram," Multimedia Tools and Applications, vol. 76, no. 6, pp. 8803–8829, Apr. 2016.

[37] R. Yao, J. Sun, Y. Zhou, and D. Chen, "Video stitching based on iterative hashing and dynamic seam-line with local context," Multimedia Tools and Applications, vol. 76, no. 11, pp. 13615–13631, Jul. 2016.

[38] M. A. Fischler and R. C. Bolles, "Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," Readings in Computer Vision, pp. 726–740, 1987.

[39] P. H. S. Torr and A. Zisserman, "MLESAC : A New Robust Estimator with Application to Estimating Image Geometry," Computer Vision and Image Understanding, vol. 78, no. 1, pp. 138–156, Apr. 2000.

[40] R. Prados, R. Garcia, and L. Neumann, "State of the Art in Image Blending Techniques," SpringerBriefs in Computer Science, pp. 35–60, 2014.

[41] Y. Xiong and K. Pulli, “Color correction based image blending for creating high resolution panoramic images on mobile devices,” ACM SIGGRAPH ASIA 2009 Posters on - SIGGRAPH ASIA ’09, 2009.

[42] J.-Y. Guillemaut, J. Kilner, J. Starck, and A. Hilton, “Dynamic feathering : minimising blending artefacts in view-dependent rendering,” IET 4th European Conference on Visual Media Production (CVMP 2007), 2007.

[43] P. J. Burt and E. H. Adelson, “A multiresolution spline with application to image mosaics,” ACM Transactions on Graphics, vol. 2, no. 4, pp. 217–236, Oct. 1983.

[44] Y. Xiong and K. Pulli, “Gradient Domain Image Blending and Implementation on Mobile Devices,” Mobile Computing, Applications, and Services, pp. 293–306, 2010.

[45] A. H.-D. Cheng and D. T. Cheng, “Heritage and early history of the boundary element method,” Engineering Analysis with Boundary Elements, vol. 29, no. 3, pp. 268–302, Mar. 2005.

[46] Jeonghwe Gu, Hangil Joe, and Son-Cheol Yu, “On preprocessing methods for feature-based photo-mosaic underwater,” OCEANS 2016 MTS/IEEE Monterey, Sep. 2016.

[47] J. Arnal and I. Mayzel, “Parallel techniques for speckle noise reduction in medical ultrasound images,” Advances in Engineering Software, vol. 148, p. 102867, Oct. 2020.

[48] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, “Bundle Adjustment — A Modern Synthesis,” Lecture Notes in Computer Science, pp. 298–372, 2000.

[49] G. M. Jacob and S. Das, “Panorama from Representative Frames of Unconstrained Videos Using DiffeoMeshes,” Lecture Notes in Computer Science, pp. 166–182, 2019.

[50] G. Messina, S. Battiato, and A. Castorina, “Exposure Correction for Imaging Devices,” Image Processing Series, pp. 323–349, Sep. 2008.

[51] F. W. Cathey and D. J. Dailey, “Mathematical theory of image straightening with applications to camera calibration,” 2006 IEEE Intelligent Transportation Systems Conference, 2006.

[52] O. T. Tursun, A. O. Akyüz, A. Erdem, and E. Erdem, “The State of the Art in HDR Deghosting : A Survey and Evaluation,” Computer Graphics Forum, vol. 34, no. 2, pp. 683–707, May 2015.

[53] M.-L. Duplaquet, “Building large image mosaics with invisible seam lines,” Visual Information Processing VII, Jul. 1998.

[54] A. Mills and G. Dudek, “Image stitching with dynamic elements,” Image and Vision Computing, vol. 27, no. 10, pp. 1593–1602, Sep. 2009.

[55] L. Zeng, S. Zhang, J. Zhang, and Y. Zhang, “Dynamic image mosaic via SIFT and dynamic programming,” Machine Vision and Applications, vol. 25, no. 5, pp. 1271–1282, Oct. 2013.

[56] L. Li, J. Yao, H. Li, M. Xia, and W. Zhang, “Optimal seamline detection in dynamic scenes via graph cuts for image mosaicking,” Machine Vision and Applications, vol. 28, no. 8, pp. 819–837, Sep. 2017.

[57] D. Comaniciu and P. Meer, “Mean shift : a robust approach toward feature space analysis,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 603–619, May 2002.

[58] S. Saito, R. Arai, and Y. Aoki, “Seamline Determination Based on Semantic Segmentation for Aerial Image Mosaicking,” IEEE Access, vol. 3, pp. 2847–2856, 2015.

[59] Q. Dong and J. Liu, “Seamline Determination Based on PKGC Segmentation for Remote Sensing Image Mosaicking,” Sensors, vol. 17, no. 8, p. 1721, Jul. 2017.

[60] M. Kerschner, “Seamline detection in colour orthoimage mosaicking by use of twin snakes,” ISPRS Journal of Photogrammetry and Remote Sensing, vol. 56, no. 1, pp. 53–64, Jun. 2001.

[61] L. Li, J. Tu, Y. Gong, J. Yao, and J. Li, “Seamline network generation based on foreground segmentation for orthoimage mosaicking,” ISPRS Journal of Photogrammetry and Remote Sensing, vol. 148, pp. 41–53, Feb. 2019.

[62] M. Uyttendaele, A. Eden, and R. Skeliski, “Eliminating ghosting and exposure artifacts in image mosaics,” Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.

[63] Z. Qu, T. Wang, S. An, and L. Liu, “Image seamless stitching and straightening based on the image block,” IET Image Processing, vol. 12, no. 8, pp. 1361–1369, Aug. 2018.

[64] Qi Zhi and J. R. Cooperstock, “Depth-based image mosaicing for both static and dynamic scenes,” 19th International Conference on Pattern Recognition, Dec. 2008.

[65] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image Quality Assessment : From Error Visibility to Structural Similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[66] H. S. Qureshi, M. M. Khan, R. Hafiz, Y. Cho, and J. Cha, “Quantitative quality assessment of stitched panoramic images,” IET Image Processing, vol. 6, no. 9, pp. 1348–1358, Dec. 2012.

[67] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment,” The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003.

[68] F. A. Kruse, A. B. Lefkoff, J. W. Boardman, K. B. Heidebrecht, A. T. Shapiro, P. J. Barloon, and A. F. H. Goetz, “The spectral image processing system (SIPS)-interactive visualization and analysis of imaging spectrometer data,” AIP Conference Proceedings, 1993.

[69] “Gain (Magnitude Ratio),” Van Nostrand’s Scientific Encyclopedia, Oct. 2005.

[70] S. Laaroussi, A. Baataoui, A. Halli, and K. Satori, “Quantitative measure for mosaics with dynamic elements using dense optical flow and the structure similarity index measure.,” 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), Apr. 2020.

[71] S. Laaroussi, A. Baataoui, A. Halli, and K. Satori, “Dynamic mosaicking : region-based method using edge detection for an optimal seamline,” Multimedia Tools and Applications, vol. 78, no. 16, pp. 23225–23253, May 2019.

[72] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, “Color transfer between images,” IEEE Computer Graphics and Applications, vol. 21, no. 4, pp. 34–41, 2001.

[73] J. CANNY, “A Computational Approach to Edge Detection,” Readings in Computer Vision, pp. 184–203, 1987.

[74] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.

[75] W. Xu and J. Mulligan, “Performance evaluation of color correction approaches for automatic multi-view image and video stitching,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Jun. 2010.

[76] S. V. Burtsev and Y. P. Kuzmin, “An efficient flood-filling algorithm,” Computers & Graphics, vol. 17, no. 5, pp. 549–561, Sep. 1993.

[77] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, “The 2017 DAVIS Challenge on Video Object Segmentation,” arXiv :1704.00675, 2017

[78] S. Laaroussi, A. Baataoui, A. Halli, and K. Satori, “Dynamic mosaicking : combining A* algorithm with fractional Brownian motion for an optimal seamline detection,” IET Image Processing, vol. 14, no. 13, pp. 3169–3180, Nov. 2020.

[79] B. B. Mandelbrot, “Fractional Brownian Motions and Fractional Gaussian Noises,” Encyclopedia of Statistical Sciences, Oct. 2004.

[80] P. Tahmasebi, A. Hezarkhani, and M. Sahimi, “Multiple-point geostatistical modeling based on the cross-correlation functions,” Computational Geosciences, vol. 16, no. 3, pp. 779–797, Mar. 2012.

[81] O. Wink, W. J. Niessen, and M. A. Viergever, “Minimum cost path determination using a simple heuristic function,” Proceedings 15th International Conference on Pattern Recognition. ICPR-2000.

[82] J.-F. Coeurjolly, “Simulation and Identification of the Fractional Brownian Motion : A Bibliographical and Comparative Study,” Journal of Statistical Software, vol. 5, no. 7, 2000.

[83] A. T. A. Wood and G. Chan, “Simulation of Stationary Gaussian Processes in [0,1] d,” Journal of Computational and Graphical Statistics, vol. 3, no. 4, p. 409, Dec. 1994.

[84] N. Dhanachandra, K. Manglem, and Y. J. Chanu, “Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm,” Procedia Computer Science, vol. 54, pp. 764–771, 2015.

[85] S. N. Mohanty, J. M. Chatterjee, S. Jain, A. A. Elngar, and P. Gupta, Eds., “Recommender System with Machine Learning and Artificial Intelligence,” Jun. 2020.

[86] G. Farnebäck, “Two-Frame Motion Estimation Based on Polynomial Expansion,” Lecture Notes in Computer Science, pp. 363–370, 2003.

Publications scientifiques

Articles publiés

- S. Laaroussi, A. Baataoui, A. Halli, and K. Satori, "Dynamic mosaicking : region-based method using edge detection for an optimal seamline," Multimedia Tools and Applications, vol. 78, no. 16, pp. 23225–23253, May 2019.
- S. Laaroussi, A. Baataoui, A. Halli, and K. Satori, "Dynamic mosaicking : combining A* algorithm with fractional Brownian motion for an optimal seamline detection," IET Image Processing, vol. 14, no. 13, pp. 3169–3180, Nov. 2020.
- S. Laaroussi, A. Baataoui, A. Halli, and K. Satori, "Item-to-item recommender system with simultaneous use of multiple images for image mosaicking creation in dynamic scenes," IET Image Processing, vol. 14, no. 17, pp. 4726–4735, Dec. 2020.

Communications

- S. Laaroussi, A. Baataoui, A. Halli, and S. Khalid, "A dynamic mosaicking method based on histogram equalization for an improved seamline," Procedia Computer Science, The first International Conference on intelligent computing in Data Sciences, ICDS2017, Meknes, vol. 127, pp. 344–352, 2018.
- S. Laaroussi, A. Baataoui, A. Halli, and S. Khalid, "A dynamic mosaicking method for finding an optimal seamline with Canny edge detector," Procedia Computer Science, The second International Conference on intelligent computing in Data Sciences, ICDS2018, Fez, vol. 148, pp. 618–626, 2019.
- S. Laaroussi, A. Baataoui, A. Halli, and K. Satori, "Quantitative measure for mosaics with dynamic elements using dense optical flow and the structure similarity index measure.," 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), Meknes, Apr. 2020.