TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ETS	École de technologie supérieure
DM	Carte de profondeur (*Depth Map*)
SfM	Structure par le mouvement (*Structure from Motion*)
SP	Superpixels
MRF	Champ aléatoire de Markov (*Markov Random Field*)
SIFT	*Scale-Invariant Feature Transform*
BM	Correspondance de blocs *Block Matching*
AEE	Erreur de position finale moyenne *Average Endpoint Error*
AAE	Erreur angulaire moyenne *Average Angular Error*

INTRODUCTION

L'intérêt du public envers le visionnement de contenu 3D est en croissance à l'heure actuelle. Cela est probablement relié à la nouvelle facilité et à l'amélioration des médias servant à visualiser du contenu 3D à la maison. Depuis quelques années, les écrans se sont améliorés en affichant une meilleure résolution et à une fréquence de rafraîchissement plus haute, ce qui permet à de nouvelles méthodes moins dispendieuses d'apparaître pour permettre de visualiser du contenu 3D. Les salles de cinéma sont presque toutes maintenant équipées pour la projection 3D et plusieurs des nouveaux films produits par Hollywood sont projetés en 3D. Il ne faut pas négliger aussi la recrudescence des nouvelles technologies liées à la réalité augmentée ou virtuelle par exemple comme l'oculus rift et le HTC Vive qui augmentent aussi l'intérêt pour des produits disponibles au grand public. Le 3D devenant de plus en plus répandu, la demande pour du contenu stéréoscopique augmente.

Cependant, les techniques pour produire du contenu stéréoscopique sont encore en général assez coûteuses et nécessitent du traitement additionnel pour arriver à des résultats acceptables pour la diffusion de 3D. La méthode la plus évidente est de filmer une scène avec deux caméras montées sur un support prévu à cet effet. Cela augmente beaucoup les coûts de production d'un film, car tout le traitement qui était effectué sur une seule « pellicule » doit maintenant être effectué sur deux pellicules. D'autant plus que ces caméras doivent être synchronisées avec beaucoup de précision pour capter les mêmes images au même moment de deux points de vue différents. C'est pourquoi des moyens plus simples et moins coûteux sont recherchés. Une méthode alternative et moins coûteuse est la conversion de contenu 2D à 3D. Certains studios de cinéma emploient cette technique et procèdent de différentes façons pour arriver à la conversion. La méthode la plus utilisée nécessite l'intervention humaine pour estimer la profondeur des objets dans chaque scène. Cette technique utilise ce que l'on appelle une carte de profondeur ou *depth map (DM)*. Cette *DM* est une image en tons de gris représentant la distance relative entre la caméra et les objets dans la scène. Cette image est utilisée pour convertir

l'image couleur qui a été captée en une paire d'images stéréoscopiques pour être projetée en 3D. La conversion peut être faite par des artistes qui vont dessiner la profondeur de la scène sur les images ou par des outils qui tentent d'extraire l'information de profondeur dans la scène à partir de l'image originale.

Ce mémoire documente et explique la recherche effectuée pour le développement d'une technique ayant comme but : 1) la génération semi-automatique de cartes de profondeurs à partir d'une séquence d'images, 2) l'estimation automatique du mouvement effectuée sur une séquence d'images. Ces deux algorithmes ont été développés en utilisant des techniques semblables basées sur le calcul de la similarité entre les pixels d'images consécutives dans une séquence vidéo. La technique utilisée dans les deux cas est basée sur un modèle probabiliste de propagation de l'information utilisant un algorithme de « Random Walker » Grady (2006).

Les cartes de profondeur sont utilisées dans plusieurs algorithmes de conversion 2D à 3D car elles permettent de fournir de l'information sur la profondeur des objets. Elles sont souvent créées manuellement par des artistes spécialisés qui peuvent annoter un nombre élevé d'images, par exemple pour un film 2D que l'on voudrait convertir en 3D. Ce processus est souvent long et coûteux faute de solution universellement adoptée et pourrait être amélioré. La génération semi-automatique de cartes de profondeur veut fournir un processus plus rapide de conversion de contenu existant en 2D vers du 3D. La technique proposée a pour but de permettre à un utilisateur d'appliquer un nombre minime d'annotations sur certaines images clés d'une séquence d'images. Ces annotations représentent la profondeur relative des objets dans la scène et consistent en seulement quelques traits rapides sur les objets dans l'image clé.

Cette image est par la suite segmentée pour propager l'information de profondeur contenue dans les annotations dans toute l'image pour attribuer à chaque pixel de l'image une valeur de profondeur relative. Une fois ce processus terminé pour une image, il faut propager cette information à travers toute la séquence d'images pour que l'annotation d'une seule image permette

de trouver la profondeur sur toutes les images de la séquence. Cela doit être simple à utiliser, assez rapide pour permettre de sauver du temps par rapport aux méthodes manuelles utilisées et doit produire des résultats plus précis que les méthodes automatiques qui existent.

Durant la recherche pour réaliser cet algorithme, il a été observé que la logique utilisée pour la propagation des annotations à travers la séquence d'image pouvait aussi être appliquée pour résoudre un autre problème, celui de l'estimation de mouvement. En fait, ces deux problèmes sont partiellement reliés car pour arriver à propager correctement l'information de l'image clé aux autres images de la séquence, il faut trouver le déplacement de chaque pixel à travers le temps et cela est fait en estimant le mouvement entre deux images. Un algorithme a été développé qui, de façon automatique, analyse les différences entre deux images et trouve le déplacement le plus probable effectué par chaque pixel.

C'est en majorité sur la technique d'estimation de mouvement que la recherche a été effectuée puisque cette technique est nécessaire à la méthode proposée pour la génération de *DM* et que l'idée générale de l'algorithme est partagée par cette dernière. Une technique efficace d'estimation de mouvement améliore les résultats prévus de la technique de génération de *DM*. Le problème de retrouver un pixel correspondant dans deux images est un problème mal défini pour lequel il n'existe pas encore de méthode entièrement satisfaisante. En effet, il y a plusieurs problèmes qui n'ont pas de réponse bien définie dans cette situation. Par exemple, dans le cas où il y a des changements d'illuminations entre les deux images, le même pixel sera "différent" et donc très difficile à retrouver d'une image à l'autre. Aussi, dans un cas qui nous concerne, en présence de mouvement entre les deux images, certains pixels ne trouveront pas de correspondance puisque certains objets pourraient avoir bougé, en obstruer d'autres ou d'autres pourraient se déformer. Plusieurs autres difficultés peuvent être présentes dans notre cas, comme l'absence de textures permettant de différencier un pixel d'un autre pixel dans une zone faiblement texturée, l'apparition d'occlusions dues au mouvement, les changements

d'illumination, le flou relié au mouvement et plusieurs autres. La méthode proposée utilise certaines heuristiques qui seront décrites plus loin pour essayer de combler ces difficultés.

CHAPITRE 1

ÉTAT DE L'ART

1.1 Génération d'une carte de profondeur

Une carte de profondeur *depth map* (DM) est une image composée de niveaux de gris où la valeur de gris ou d'intensité attribuée à un pixel représente la profondeur relative, généralement par rapport au plan de la caméra. Pour générer une carte de profondeur à partir d'une séquence vidéo il faut donc attribuer à chaque pixel dans chaque image de la séquence une valeur correspondant à la profondeur relative entre les pixels. Ce processus nécessite d'extraire de l'information dans l'image et de l'interpréter pour attribuer une valeur de profondeur aux pixels. À moins d'avoir de l'information spécifique sur la caméra et sur la lentille utilisée pour filmer la scène, la profondeur extraite ne représentera pas la profondeur réelle des objets, mais plutôt une indication sur l'ordre des objets par rapport à la caméra. Il y a plusieurs méthodes pour obtenir l'information liée à la profondeur dans l'image, certaines sont basées sur le fonctionnement du système visuel humain, certaines utilisent des techniques géométriques ou mathématiques. Dans une situation réelle, l'information de profondeur n'est évidemment pas entièrement présente dans l'image alors il faut souvent avoir recours à des techniques pour remplir les zones où l'information est manquante. Cette section présente une classification des différentes méthodes étudiées selon la source d'information utilisée pour extraire de l'information sur la profondeur.

1.1.1 Système visuel humain

Avant d'essayer de convertir du contenu 2D en du contenu 3D il est important de comprendre comment fonctionne le système visuel humain, autant pour percevoir la profondeur dans une scène que pour comprendre comment il est possible de tromper le cerveau en lui projetant des images lui faisant croire qu'il voit de la profondeur. La perception de la profondeur est reliée à deux facteurs, le premier vient du fait que l'humain voit des images avec deux yeux,

le cerveau reçoit donc deux images de la même scène qui sont prises de deux points de vue et d'orientations différents. On appelle cela la vision binoculaire. Le second vient de l'apprentissage subconscient dès l'enfance à reconnaître certains signes dans une image comme étant des indicateurs de profondeur que l'on peut classifier comme vision monoculaire. Comme il sera expliqué dans les sections suivantes, pour permettre d'estimer la profondeur dans une image avec un algorithme, il faut utiliser aussi ces facteurs pour détecter et exploiter les signes présents dans une ou plusieurs images (Vazquez *et al.* (2013)).

Les indices binoculaires, comme leur nom l'indique, nécessitent d'utiliser deux images pour percevoir la différence entre les deux points de vue de la même scène. En ayant deux images de la même scène avec une petite différence de point de vue entre elles, on peut calculer la disparité entre les deux points de vue. C'est-à-dire la différence de position entre le même point de la scène projeté dans les deux images. À partir de cette disparité, de la distance avec la caméra et de la distance entre les deux caméras, il est possible de calculer la profondeur des objets dans la scène. On appelle cette caractéristique la stéréopsie.

Les signes monoculaires sont plus facilement exploitables puisqu'ils ne nécessitent pas d'avoir plusieurs caméras ou plusieurs images de la même scène pour discerner la profondeur. Parmi ces indices de profondeur, on retrouve des indices qui peuvent sembler simplistes pour l'œil humain, mais qui sont parfois difficiles à traduire en instructions pour un algorithme. Par exemple, en regardant une table, nous sommes capables de percevoir sa distance approximative car chaque personne a une bonne idée de la grandeur générale d'une table. Si la table apparaît comme petite, elle doit être éloignée, si la table apparaît comme plus grande, elle doit être proche. Ceci est plus difficile à traduire en chiffres pour inclure cette intuition dans un algorithme d'estimation de la profondeur. D'autres indices sont : la perspective linéaire, la taille relative des objets, le mouvement, l'ordre des objets (si un objet cache partiellement un autre objet, il doit forcément être devant) et même la couleur, l'illumination et le focus (Vazquez *et al.* (2013)). Il en existe plusieurs autres, mais en général ce sont ces indices qui sont exploités par les méthodes d'estimation de la profondeur dans une image. Les sections suivantes présentent plusieurs méthodes selon le ou les indices de profondeur qu'elles utilisent.

1.1.2 Techniques d'estimation de la profondeur automatiques

Les méthodes fonctionnant sans l'intervention d'un utilisateur sont dites des méthodes automatiques. Elles se basent entièrement sur l'information extraite des images et sur des heuristiques pour estimer la profondeur dans une image. Pour arriver à estimer la profondeur des pixels dans une séquence d'images, certaines méthodes existent qui utilisent les mêmes indices visuels que l'humain. C'est-à-dire qu'elles vont détecter certaines particularités de l'image et les interpréter comme le ferait un humain pour percevoir la profondeur. Ces méthodes sont généralement utilisées pour obtenir un résultat adéquat rapidement puisqu'aucune intervention d'un utilisateur n'est requise. Souvent, les limitations de ces méthodes sont reliées à la complexité de calcul qui a un impact important sur le temps de calcul. Pour conserver une rapidité de calcul assez élevée, il faut parfois utiliser des heuristiques qui permettent de sauver une partie des calculs.

Pour qu'un humain perçoive de la profondeur d'une scène, une combinaison des indices visuels présentés plus haut est utilisée, mais pour un programme la tâche est plus complexe. Arriver à utiliser l'information de profondeur d'un seul indice visuel implique des calculs devenant rapidement très complexes et cela peut prendre un temps important à calculer. Cela force généralement les algorithmes à choisir un nombre limité d'indices visuels pour trouver la profondeur, ce qui limite souvent leur capacité à s'adapter à tous les cas possibles. Étant donné que tous les indices visuels ne se prêtent pas à toutes les situations, il est possible, par exemple de catégoriser le contenu d'une scène avant de décider quelle méthode utiliser. Certaines méthodes vont effectuer une analyse préalable de la scène, tandis que d'autres vont se limiter à fonctionner seulement dans un ou plusieurs cas précis. Les sections suivantes présentent des publications de ces méthodes qui ont été étudiées pour analyser l'état courant des solutions existantes à l'estimation de profondeur dans une scène, séparées selon l'indice visuel utilisé.

1.1.2.1 Profondeur par le mouvement

Un des indices visuels utilisés par l'humain pour percevoir de la profondeur vient du déplacement des objets dans la scène ou, plus simplement, du mouvement. Les techniques d'extraction de la profondeur utilisant le mouvement se basent généralement sur la prémisse que les objets qui se déplacent le plus rapidement sont en avant plan dans la scène par opposition aux objets plus éloignés qui auront peu ou pas de mouvement à travers la séquence d'images. Par exemple, dans le cas où une caméra serait en mouvement devant une scène fixe, ce qui est le plus loin en arrière plan bougera lentement par rapport à ce qui se trouve en premier plan dans l'image. Il est important de noter que des objets pouvant en occlure d'autres dans une image fournissent aussi de l'information de profondeur, car si un objet en déplacement passe devant un autre objet, on peut en conclure que le premier objet est en avant du second (Liu And Christopher (2012)).

Une famille de techniques regroupant plusieurs méthodes pour arriver à extraire la profondeur d'une scène à partir du mouvement se nomme *Structure from Motion* (SfM) (Ward *et al.* (2011)) (Cheng *et al.* (2010)). L'idée générale derrière SfM est de tenter de calculer les paramètres de caméra (la translation, la rotation et la matrice intrinsèque) à partir du déplacement des objets dans une scène sur plusieurs images consécutives. Cette façon de procéder permet, une fois les paramètres de caméra établis, de calculer avec beaucoup de précision la profondeur réelle de chaque pixel dans l'image. Il est important de noter que l'on parle bien ici de la profondeur réelle par opposition à la profondeur relative qui est calculée par les autres techniques. Il est possible d'inverser le calcul de la projection d'un point dans l'espace sur le plan de la caméra en connaissant ses paramètres et donc de calculer la profondeur réelle de chaque pixel au lieu d'une estimation. Il est aussi nécessaire de posséder l'information sur la caméra utilisée ainsi que sur la lentille pour pouvoir appliquer toute transformation oculaire dans les calculs de profondeur.

Certains travaux utilisent la géométrie épipolaire pour établir les paramètres de caméra et reconstruire des matrices de projection qui font la correspondance entre la projection des

points affichée dans l'image et la position des points dans un espace en trois dimensions (Knorr And Sikora (2006)). Ces méthodes utilisent des contraintes calculées à l'aide de la géométrie épipolaire et de la SfM pour trouver le même pixel d'une image à l'autre et pour détecter des pixels qui seraient en occlusion (Zhang *et al.* (2009)).

SfM permet d'obtenir une grande précision lors du calcul de la profondeur de pixel dans une image. Cependant, ces techniques imposent certaines contraintes. Le calcul des paramètres de caméra doit généralement provenir d'une scène fixe où il y a, au moins sur quelques images rapprochées dans la séquence, un mouvement de caméra continu. En d'autres mots, pour effectuer un calcul précis des paramètres de caméra, on assume que la caméra doit bouger par rapport à une scène fixe. Il est donc préférable de vérifier au préalable si la scène se prête bien au SfM ou sinon d'utiliser une autre méthode.

1.1.2.2 Profondeur par le focus/defocus

Selon le fonctionnement d'une caméra, on sait que plus un objet est éloigné du plan focus de la lentille, plus cet objet apparaîtra comme flou à l'observateur. Ce phénomène peut être utilisé pour estimer la profondeur des objets dans la scène. L'idée générale est d'arriver à déterminer le degré de flou pour chaque pixel et par la suite d'attribuer une valeur de profondeur en fonction du degré de ce flou. Les techniques utilisant le flou comme source d'information nécessitent généralement plusieurs images prises de la même scène avec des distances focales différentes. En trouvant l'image où un pixel est le plus au focus, il est possible d'extrapoler sa distance par rapport à la caméra (Malik *et al.* (2007)) (Rajan And Chaudhuri (2003)) (Lee *et al.* (2013)). Le problème principal de ces méthodes est qu'un objet se trouvant à une certaine distance en avant du plan focal et un autre se trouvant en arrière du plan focal à la même distance aura un degré de flou similaire. Il est difficile de savoir si un objet est en avant ou en arrière d'un autre en constatant seulement qu'il est flou dans une image. Seuls les objets qui sont au focus peuvent se voir attribuer une profondeur avec confiance lorsqu'on connait la distance focale.

La famille de méthodes utilisant le focus d'une caméra se nomme *Shape from Focus* (SfF) dans la littérature. Comme pour les techniques de SfM, de l'information sur les paramètres de caméra (dans ce cas, les particularités de la lentille) est nécessaire pour l'application de la méthode et permet une reconstruction beaucoup plus fidèle de la profondeur dans une image. En mettant en relation les informations de la lentille et l'image au focus de chaque pixel, il est possible de recréer un nuage de points qui représentent la scène. À partir de ce nuage de points, il est possible de reconstruire la profondeur dans une scène. Pour attribuer une valeur de focus aux pixels d'une image, il est nécessaire d'utiliser un opérateur pour mesurer le focus et plusieurs opérateurs différents sont proposés dans la littérature comme l'opérateur SUSAN (Mendapara *et al.* (2009)), comme une estimation basée sur les ondelettes (Mendapara (2010)) et/ou une analyse du domaine fréquentiel de l'image (Lin *et al.* (2010)).

Cette façon de procéder requiert malheureusement plusieurs prises de vue de la même scène avec différentes distances focales pour faire varier la distance de focus. Encore une fois, une scène en mouvement ne se prête pas très bien au processus de SfF. Il n'est pas toujours possible de prendre plusieurs prises de vue de la même scène et donc le résultat obtenu dépend fortement du contenu à analyser. Certains auteurs ont par contre suggéré l'utilisation d'un ensemble de caméras ayant des longueurs focales différentes. Ces caméras filment la même scène, mais en ayant différentes longueurs focales, on peut substituer ceci à prendre plusieurs prises de vue de la même scène.

1.1.2.3 Profondeur par la géométrie de la scène

Il est parfois possible de guider l'estimation de profondeur dans une image par la géométrie générale de la scène. Pour ce faire, un algorithme fonctionne un peu comme un humain car on peut extraire des structures géométriques reconnaissables dans la scène puis leur assigner une profondeur selon leur structure. Par exemple, dans une image, on peut parfois trouver des points de fuite. Ces points représentent l'endroit où plusieurs lignes se rejoignent, par exemple, deux rails de chemin de fer qui s'éloignent ne sont pas parallèles dans l'image, mais se rejoignent à un point éloigné. En utilisant ce genre d'information, on peut estimer que la profon-

deur de ces structures augmente en s'éloignant de la caméra. C'est l'information utilisée par (Han And Hong (2011)) qui, combinée avec une segmentation par l'intensité, permet d'obtenir une estimation de la profondeur de la scène.

Certaines techniques sont basées sur l'analyse du contenu de la scène pour arriver à trouver ces indices et à en extraire la profondeur. Dans certains cas, on utilise un opérateur de traitement d'image, comme l'opérateur Sobel par exemple, pour extraire toutes les lignes dans l'image (Freiling And Schumann (2013)) ou l'opérateur Haar pour trouver les coins (Bolecek And Ricny (2013)). On cherche ensuite à trouver des points de fuite ou des plans dans l'image. Par exemple, on pourrait chercher la ligne de l'horizon ou chercher les limites des murs, plafonds et planchers d'une pièce. Une fois cette information extraite, on peut plus facilement aller assigner des valeurs de profondeur à ces structures (Lee *et al.* (2013)). Par exemple, un dégradé du plus proche au plus loin appliqué au plancher d'une pièce. Ensuite tous les objets qui entrent en contact avec ce plancher vont avoir comme profondeur la valeur qui a été attribuée à l'endroit de contact (Deng And Jiang (2008)) (Cigla And Alatan (2009)).

Ces techniques fonctionnent bien pour avoir une première estimation de la profondeur dans une image. Les résultats sont souvent trop rigides pour que ces techniques soient utilisées seules pour l'estimation de la profondeur. Par rigides, on entend ici que souvent les objets se feront attribuer une seule valeur de profondeur ce qui signifie qu'un objet sera représenté par un seul plan de profondeur. Par exemple, si une personne est proche de la caméra, on voudra voir des détails de profondeur sur celle-ci et pas seulement un plan de profondeur. Il faut généralement raffiner les résultats de ces méthodes, mais elles fournissent un bon début pour une image avec du contenu ayant des structures géométriques faciles à identifier.

1.1.2.4 Profondeur par la couleur ou la réflexion

Un autre indice visuel que l'on peut utiliser pour estimer la profondeur est la couleur dans l'image. Certaines particularités de la transmission de la couleur du spectre visuel font que les couleurs chaudes sont généralement plus présentes à courtes distances. Par opposition, on

peut aussi supposer que les tons de bleu ou le gris sont plus associés au ciel ou à des pixels plus éloignés de la caméra. Certaines méthodes ont choisi de fonctionner avec la couleur pour fournir une première approximation de profondeur pour une image. On peut par exemple en utilisant les niveaux de chroma rouge provenant du modèle de couleur **YCbCr** assumer que plus l'image comporte du rouge, plus l'objet est près de la caméra. Tam *et al.* (2009) sont arrivés à des résultats concluants mentionnant que même avec une DM imparfaite on arrivait à générer du contenu stéréoscopique satisfaisant. Ils en ont tiré la conclusion que le cerveau humain effectue une grande partie du travail quant à la visualisation 3D et il n'a pas besoin que le 3D soit parfait pour arriver à voir de la profondeur.

Plusieurs autres méthodes se servent simplement de la couleur dans l'image ou dans plusieurs images pour établir la stéréo-correspondance entre deux images. Avec cette correspondance, ils parviennent à calculer une disparité, par exemple créée par le déplacement de la caméra autour d'une scène statique ou de multiples prises de vues comme dans le cas d'un ensemble de deux caméras. En connaissant la distance entre les prises de vues et avec la disparité calculée, il est possible d'estimer la profondeur dans une image (Lee *et al.* (2008)). Les différences de points de vue de caméra n'ont pas à être très importantes pour trouver la profondeur dans une scène rapprochée. Une publication intéressante suggère l'utilisation d'un objectif de caméra modifié qui produit un décalage minime entre trois filtres (rouge, vert et bleu). En connaissant la distance entre ces ouvertures d'objectif et en calculant la disparité dans l'image résultante, ils arrivent à estimer la profondeur (Lee *et al.* (2012)).

La réflexion peut aussi jouer un rôle important dans l'estimation de la profondeur. Si l'on se trouve en présence de plusieurs images de la même scène, mais sous des illuminations différentes, on peut estimer les vecteurs normaux des surfaces dans la scène. Avec ces normales, il est possible de déduire la géométrie des objets présents dans la scène. Ces techniques sont souvent très sensibles au bruit dans les images ce qui les rend difficilement applicables hors d'un environnement contrôlé. Il faut donc employer des techniques qui tiennent compte du bruit (Harrison And Joseph (2012)).

Les méthodes se servant de la couleur ont plus pour but de fournir un bon point de départ à l'estimation de profondeur en étant jumelées avec d'autres méthodes que de fournir un résultat parfait. En effet, on pourrait facilement combiner l'information provenant de la couleur avec l'information provenant de la géométrie de la scène. On augmente beaucoup la certitude, par exemple, que l'on est en présence d'une scène avec un ciel si on trouve une ligne d'horizon et que ce qui est au-dessus de cette ligne présente des tons de bleu.

1.1.2.5 Profondeur par apprentissage (reconnaissance d'objets et de scène)

Comme l'humain, il est possible d'estimer la profondeur en reconnaissant certains objets dans une scène. Par exemple, il est assez aisé de reconnaître un ciel dans une image et il en va de même pour le sol. Dans le cas d'un ciel par exemple, la profondeur attribuée serait 0 et pour un sol, on aurait un dégradé de plus en plus profond du bas de l'image vers l'horizon. Il est aussi parfois possible de reconnaître certains objets dans une scène. Il est possible de créer une librairie d'objets qui sont communément observés, par exemple dans le cas d'une voiture, un modèle de profondeur peut exister, qui représente toutes les variations de profondeurs normalement observées pour une voiture et une fois l'objet reconnu, on déforme le modèle original pour représenter le cas précis. Cela permet de fournir une structure précise pour l'estimation de profondeur et aussi de sauver beaucoup de temps puisqu'une grande partie du travail est déjà fait grâce à la réutilisation de modèles fréquents (Konrad *et al.* (2013)).

Certaines méthodes poussent plus loin cette reconnaissance de la scène en classifiant par exemple si la scène présente un objet en premier plan (par exemple si on avait filmé une personne en avant plan). Quand on se trouve dans ce cas de figure, une segmentation permet de séparer l'objet en premier plan du reste de la scène et permet de lui attribuer des valeurs de profondeur qui reflètent la situation où ce serait l'objet principal dans la scène. On peut par la suite traiter tout le reste comme étant en arrière plan puisque ce n'est pas le plus important dans la scène (Jung And Ho (2010)) (Lee *et al.* (2013)).

D'autres méthodes suivent la prémisse qu'il est possible de faire apprendre à un système à reconnaître tous les types de scènes les plus fréquentes. Ces systèmes utilisent des images 3D qui sont disponibles dans une librairie ou sur le web pour effectuer un apprentissage de la structure de profondeur associée à certaines caractéristiques dans l'image. Lorsqu'on lui présente une nouvelle image, le système cherche à retrouver des caractéristiques précises pour trouver le modèle de profondeur associé à l'image qui serait la plus similaire (Konrad *et al.* (2013)). Cela permet, dans la plupart des cas, de reproduire le modèle de profondeur appris et de l'appliquer à la nouvelle image présentée. Comme dans tout autre système à apprentissage, les résultats dépendent de la qualité de l'apprentissage et il reste une forte chance de trouver une mauvaise réponse si on se retrouve en présence d'un cas totalement nouveau (Saxena *et al.* (2009)).

1.1.3 Techniques d'estimation de la profondeur semi-automatiques

Il existe aussi d'autres méthodes qui diffèrent des précédentes parce qu'elles nécessitent l'intervention d'un utilisateur. On dit de ces méthodes qu'elles sont semi-automatiques puisqu'elles sont généralement en grande partie automatisées, mais s'appuient sur de l'information entrée par l'utilisateur pour obtenir de meilleurs résultats. Comparativement aux méthodes automatiques, certains processus peuvent parfois être délégués à l'utilisateur pour parfois réduire la possibilité d'erreur, accélérer le temps de pré traitement nécessaire ou parfois simplement résoudre des problèmes difficiles à résoudre pour une méthode entièrement automatique. Par exemple, dans le cas des occlusions, l'œil humain arrive très facilement à discerner qu'un objet est derrière un autre dans une scène, mais pour une méthode automatique il faudrait implémenter une technique de détection des occlusions pour les trouver. Il en va de même dans les zones faiblement texturées, encore une fois un utilisateur arrive facilement à discerner le contour d'un objet même s'il est très similaire aux objets avoisinants et n'a pas non plus de problème avec les textures répétitives.

Pour tirer avantage des capacités de l'œil humain à bien discerner la profondeur, les méthodes semi-automatiques mettent généralement en place un système permettant à l'utilisateur d'anno-

ter une image de la façon la plus simple possible pour garder le temps nécessaire d'annotation court. Il peut par exemple permettre à un utilisateur d'annoter une image avec des coups de pinceau en attribuant une valeur de profondeur à une gradation de couleur (par exemple avec du rouge, le rouge foncé peut signifier le plus profond et le blanc peut signifier le plus proche). La complexité de ces méthodes réside souvent dans la technique employée par la suite pour propager cette information venant des annotations à toute l'image, voir à une séquence d'images. On retrouve dans la littérature des méthodes comportant des techniques de propagation de l'information basées sur des algorithmes de segmentation comme les *graph cuts* ou des techniques *watershed* (Xu *et al.* (2012)). On retrouve aussi l'utilisation de techniques de propagation basée sur l'algorithme de *Random Walk* (Grady (2006)) qui utilise les *graph cuts* pour obtenir des frontières mieux définies pour la diffusion (Phan *et al.* (2011)).

Ces techniques offrent souvent plus de flexibilité quant aux résultats obtenus puisqu'un utilisateur connaissant bien la méthode employée peut annoter l'image en connaissance de cause pour pallier aux éventuels défauts dans la méthode. Il est aussi plus facile de créer des processus itératifs qui permettent à l'utilisateur de corriger des erreurs qui seraient survenues dans l'estimation de la profondeur tout en gardant son implication au minimum par le traitement majoritairement automatique de ces méthodes. C'est dans cette direction que les recherches de solutions se sont orientées après avoir conclut de l'analyse de la littérature que trop souvent, les méthodes automatiques rencontraient les mêmes problèmes et étaient incapables de les surmonter sans passer par des ajouts complexes à ce qui était originellement une méthode plus simple. Ces techniques, souvent, n'ont pas à se baser sur des hypothèses quant au contenu de la scène à analyser puisque l'utilisateur va servir à orienter la méthode dans la bonne direction.

1.2 Estimation du mouvement

L'estimation de mouvement est l'un des problèmes principaux en traitement d'images puisque l'obtention d'une estimation de mouvement parfaite donne beaucoup d'information sur les déplacements des pixels entre deux images, mais aussi sur la composition de la scène et même sur la profondeur des objets. Une bonne estimation du mouvement entre deux images peut

permettre entre autres : de suivre le déplacement d'un objet dans le temps, de mesurer les déformations des objets entre deux images et même de savoir si un objet est plus près de la caméra qu'un autre. Plus précisément cela permet de trouver la stéréo-correspondance entre deux images ce qui signifie de trouver avec précision un lien de correspondance entre deux pixels, l'un dans une image et le second dans l'image suivante.

Dans les applications utilisant de l'estimation de mouvement, on retrouve par exemple, des applications dans le domaine médical, de la sécurité vidéo, du militaire et l'estimation de mouvement est aussi fortement utilisée dans l'encodage vidéo. Dans plusieurs des cas, l'obtention de la stéréo-correspondance par estimation de mouvement sert à faciliter la transmission d'informations d'une image unique à plusieurs autres images similaires dans une séquence vidéo. Par exemple, pour éviter d'avoir à appliquer une technique parfois longue à exécuter sur chaque image d'une séquence, on peut l'appliquer sur une image puis propager l'information sur toute la séquence si on connait le déplacement des pixels dans le temps. Plus précisément, dans le cas de l'estimation de profondeur, cela permet d'appliquer une technique plus ou moins coûteuse pour trouver la profondeur relative des objets dans une scène puis de propager l'information tant que l'on connait le déplacement des objets dans le temps.

Il y a plusieurs techniques d'estimation de mouvement et on peut les classer selon leur priorité, certaines cherchent la précision et d'autres la rapidité d'exécution (ou la plus petite complexité de calcul). Ce sont les différentes priorités des applications utilisant l'estimation de mouvement qui vont décider du classement de la technique utilisée. Par exemple, dans le cas de l'encodage vidéo, on recherche souvent un temps d'exécution très court pour la diffusion télévisuelle impliquant de traiter plusieurs trames par secondes. Il y a d'autres applications qui nécessitent une vitesse proche du temps réel et donc cela nécessite des méthodes très rapides. Pour ces situations, on fait généralement appel aux techniques basées sur la correspondance de blocs (*block matching*) qui sacrifient la précision pour réduire la complexité de calcul en trouvant le mouvement pour des blocs dans une image au lieu de trouver le mouvement pour tous les pixels.

Dans d'autres cas, la précision est plus importante que la vitesse, on peut penser à des applications médicales où le mouvement précis est très important et la contrainte de temps moins importante, par exemple pour la reconstruction après scan pour le diagnostique (puisque ces opérations peuvent être prévues après un scan, similaire à l'étude d'une radiographie, par exemple). Dans ces cas on fait plus appel à des méthodes produisant un champ dense de mouvement, principalement le calcul du flux optique (*optical flow*). Ces méthodes cherchent à trouver le déplacement précis de chaque pixel dans l'image par rapport à une image suivante dans le temps. Finalement, pour certaines applications, il faut trouver un compromis entre le temps d'exécution et les résultats obtenus, on fera donc appel à des méthodes hybrides.

1.2.1 Calcul de la correspondance de blocs (*block matching* (BM))

L'estimation de mouvement fait partie des processus nécessaires pour les techniques les plus récentes d'encodage vidéo ; il est plus rapide d'envoyer seulement la différence entre chaque trame que d'envoyer la trame complète alors l'estimation de mouvement est utilisée pour calculer ces différences. Puisque le but premier de l'encodage vidéo est généralement d'accélérer la transmission de trames, la rapidité du processus d'encodage et de décodage est importante. Il en va donc de même pour la rapidité du processus d'estimation de mouvement. Avec les capacités de calculs des appareils courants, il n'est pas encore envisageable d'effectuer un calcul de mouvement sur chaque pixel de l'image alors la technique la plus utilisée va diviser une image en plusieurs blocs et trouver le mouvement de ces blocs au lieu d'effectuer ce processus pour tous les pixels. Le principe général d'une méthode de correspondance de blocs est le suivant : il faut trouver, dans l'image suivante, le bloc présentant le moins de distorsion avec le bloc de référence courant et trouver ceci pour tous les blocs dans l'image. La distorsion est généralement calculée avec une métrique qui représente l'ensemble des différences entre les pixels constituants d'un bloc. Une fois le bloc étant le plus similaire au bloc de référence trouvé, on calcule le vecteur de mouvement entre le bloc de référence et le bloc trouvé. On effectue cette opération pour chaque bloc dans l'image et on obtient les vecteurs de mouvement pour l'image. Cette section présente un résumé des différentes techniques de correspondance

de blocs, mais cela n'est pas le sujet principal de ce mémoire. L'étude de ces méthodes est tout de même intéressante puisque certaines des techniques de simplification et d'amélioration peuvent être utilisées pour résoudre le problème d'estimation de mouvement dense et souvent les descripteurs de blocs utilisés peuvent être transportés à une technique d'estimation dense.

La technique idéale pour la recherche de correspondance de blocs serait évidemment de comparer le bloc de référence avec tous les blocs possibles dans l'image de recherche. Cela pourrait rapidement devenir trop coûteux quand on tient compte de la contrainte temps réel qu'impose la diffusion d'émissions télévisuelles, par exemple. Il faut donc utiliser certaines stratégies pour réduire la complexité de calcul tout en s'assurant de ne pas avoir trop de dégradation d'image. Les techniques cherchant à améliorer la correspondance de blocs travaillent sur plusieurs facteurs pour simplifier la recherche. Il s'agit soit de réduire le nombre de comparaisons entre les blocs, de réduire la complexité de la comparaison entre les blocs ou de prédire la meilleure zone de recherche pour trouver le meilleur bloc (Huang *et al.* (2006)).

1.2.1.1 Importance de l'étude des différentes méthodes de correspondance de blocs

Comme mentionné précédemment, la comparaison des différentes méthodes de recherche sur la correspondance de blocs n'est pas le sujet principal de ce document. Cependant, il est intéressant d'apprendre des différentes méthodes puisque le but premier de celles-ci est la rapidité d'exécution tout en gardant une précision acceptable car, pour l'application principale du BM, l'encodage vidéo, une erreur sur la prédiction de mouvement peut être très coûteuse. Les différentes techniques d'amélioration apportées par plusieurs méthodes dans ce domaine peuvent être transposées et être appliquées sur des méthodes d'estimation de champ dense de mouvement ou plus précisément sur la recherche de correspondance des pixels qui sera un morceau clé de la méthode présentée d'estimation de DM ainsi que d'estimation de mouvement.

1.2.1.2 Réduire la comparaison de blocs

Certaines méthodes se basent sur des heuristiques de recherche pour réduire le nombre de candidats à comparer au bloc de référence. On retrouve dans cette catégorie la recherche en 3 étapes *three-step-search* (Koga *et al.* (1981)), la recherche en quatre étapes *four-step-search* (Po *et al.* (1996)), la recherche logarithmique *logarithmic-search* (Jain (1981)) ainsi que la recherche en forme de diamant *diamond search* (Tham *et al.* (1998)) et plusieurs autres. Il y a un nombre important de publications présentant des techniques cherchant à réduire le nombre de blocs à comparer. Ces méthodes font la preuve expérimentale qu'effectuer la recherche de la correspondance des blocs selon un ordre spécifique présente des résultats plus rapidement et cela permet de sauver des comparaisons à effectuer dans la zone de recherche. En général, on peut accepter que la distorsion ira en croissance plus on s'éloigne de la meilleure correspondance. Alors, en essayant seulement quelques points de recherche au départ et en continuant dans la région de la meilleure correspondance trouvée, on arrive souvent à trouver le meilleur bloc tout en ayant sauvé plusieurs correspondances que l'on n'aura pas à effectuer.

D'autres méthodes se servent plutôt de l'information temporelle pour tenter de prédire la meilleure zone de recherche pour trouver le bloc le plus semblable (Chalidabhongse And Jay Kuo (1997)). On utilise les vecteurs de mouvements du bloc et des blocs voisins trouvés pour les images précédentes pour former une estimation du mouvement du bloc de référence. Cela permet de choisir, par exemple, une zone de recherche dans l'image suivante et de trouver un point de départ pour la recherche ce qui limite souvent le nombre total de comparaisons à effectuer. Ces méthodes obtiennent généralement une bonne performance dans les cas présentant des mouvements importants puisque plusieurs autres méthodes pourraient tomber dans des minimums locaux.

On peut aussi modifier l'image de recherche pour avoir une meilleure idée des zones pouvant contenir le meilleur bloc à rechercher. Des méthodes pyramidales à multi résolutions sont employées pour d'abord obtenir une correspondance au niveau plus grossier (Tzovaras *et al.* (1994)) en diminuant grandement la taille de l'image ce qui effectivement réduit le nombre de

blocs à comparer, mais aussi diminue le nombre de petits détails qui pourraient influencer la comparaison. Une fois la meilleure correspondance trouvée au niveau le plus grossier, la solution est recherchée dans les niveaux de plus en plus détaillés en commençant par la direction trouvée au niveau précédent. On utilise généralement de deux à trois niveaux de résolution tout dépendant de la taille originale de l'image. Trop de niveaux de résolution entraînent un haut risque de tomber dans des minimums locaux puisqu'une résolution trop grossière entraîne la perte de beaucoup d'information utilisée pour la correspondance.

En prenant le problème à l'envers, au lieu de réduire le nombre de blocs à comparer, on peut parfois aussi accélérer la comparaison en éliminant rapidement certains blocs qui n'ont pas de chance d'être de bons candidats pour la correspondance. Ces méthodes procèdent à ce que l'on appelle une recherche complète rapide ou accélérée dans laquelle on emploie une technique heuristique pour valider rapidement si un nouveau bloc est un bon candidat potentiel à la comparaison ou si on peut le laisser de côté. Par exemple, Li And Salari (1995) calcule d'abord la somme des valeurs de pixels dans chaque bloc et dans le bloc de référence. Il est ensuite prouvé que si la différence entre la somme d'un bloc et la somme du bloc de référence est supérieure à la meilleure somme des différences absolues trouvée jusqu'à maintenant, le bloc peut être éliminé comme candidat potentiel. Ces méthodes parviennent à diminuer le nombre de pleines comparaisons de blocs à faire ce qui accélère le temps de traitement global.

1.2.1.3 Réduire la complexité de la comparaison

Une autre façon d'accélérer la recherche de la correspondance des blocs est de réduire la complexité de la comparaison, au lieu de limiter le nombre de blocs à comparer. Typiquement la différence entre deux blocs est calculée à l'aide de la somme de la différence absolue de tous les pixels contenus dans un bloc pour obtenir un facteur de similarité. Ceci implique d'effectuer un grand nombre d'opérations pour chaque bloc ce qui peut prendre un temps important. Alors certaines méthodes se sont penchées sur des techniques qui permettraient de ne pas avoir à calculer la différence de chaque pixel pour chaque bloc ou d'accélérer cette comparaison. Par exemple, il est possible de ne comparer qu'un pixel sur deux horizontalement et vertica-

lement ce qui réduit déjà la complexité par un facteur de quatre ce que l'on appelle du sous-échantillonnage *subsampling*. Puis des techniques plus poussées se sont mises à analyser les blocs pour ne comparer que les pixels qui fournissaient l'information la plus significative ou à utiliser des méthodes de simplification des calculs pour ne calculer que la différence la plus importante dans un bloc au lieu de la somme des différences Chen *et al.* (1995).

Une autre façon d'aborder le problème est de directement réduire la quantité d'information à comparer pour chaque pixel. Typiquement, la valeur de chaque pixel est stockée sur 8 bits et il a été démontré qu'il était possible de tronquer jusqu'à 4 bits, les moins significatifs, pour réduire la quantité d'information à comparer. Le fait de stocker l'information sur moins de bits va réduire considérablement la complexité de chaque comparaison et donc l'accélérer puisqu'il y aura moins de calculs à effectuer. Ces techniques sont moins populaires car on atteint généralement rapidement un point où réduire davantage l'information entraînerait une perte de précision dans la comparaison qui pourrait mener à des erreurs d'estimation du mouvement. D'autres méthodes peuvent aussi utiliser un espace de couleurs différent du RGB traditionnel pour séparer par exemple l'information d'intensité à celle de la couleur et la comparaison peut alors n'être qu'à effectuer sur moins de caractéristiques.

1.2.2 Calcul de champ dense de mouvement (*flux optique*)

Les méthodes utilisées par des applications nécessitant une grande précision cherchent généralement un champ dense de mouvement entre deux images. C'est-à-dire qu'elles ont pour objectif d'assigner, à chaque pixel d'une image (I_t) un vecteur de mouvement $v(x,y)$ qui correspond au déplacement de ce pixel pour arriver à sa nouvelle position dans la prochaine image de la séquence ($I_{t+\Delta t}$). Pour y arriver, on procède généralement à une recherche dans une région de pixels autour de la position originale pour trouver le pixel étant le plus semblable à celui que l'on analyse. Cette recherche peut être mécanique en comparant chaque pixel, mais plus souvent, il suffit de calculer le gradient local à chaque pixel $\delta(x,y,t)$. Évidemment dans un cas optimal, cette zone de recherche serait l'image entière, mais concrètement cela est impossible quand on est en présence d'images de grandes tailles. Une image en haute définition

(1920x1080) contient plus de 2 millions de pixels, et cela serait impossible de comparer chaque pixel avec les 2 millions de pixels de l'image suivante. Cela veut dire que la taille de la zone de recherche pour la comparaison a une grande influence autant sur la qualité des résultats de chaque méthode, mais aussi sur son temps de calcul. Cette façon de procéder assume deux situations qui sont implicites à cette recherche, en premier que les pixels sont similaires d'une image à l'autre (qu'il n'y ait pas eu trop de variation d'intensité ou de couleur entre les deux images) et l'autre, indirectement, qu'un pixel n'ait pas quitté la zone de recherche (dont la grandeur est à la discrétion de la méthode employée) ou même l'image (soit sorti du cadre, ou occlus par un autre objet).

La famille de méthodes basées sur le flux optique a comme objectif de trouver le mouvement de chaque pixel entre deux images le plus précisément possible. On retrouve certaines méthodes qui obtiennent même une précision plus fine qu'un pixel (Chan *et al.* (2010)). Les méthodes de calcul du flux optique produisent généralement un champ dense de vecteurs de mouvements attachés à chaque pixel de l'image. On pourrait représenter visuellement le processus comme une grille ayant comme noeud chaque pixel de l'image et on tenterait de déformer cette grille pour représenter le mouvement des pixels d'une image à l'autre, en tirant sur chaque maille dans la direction la plus probable du mouvement. Souvent on retrouve dans ces méthodes un processus d'optimisation d'un modèle mathématique qui cherche à trouver la plus forte probabilité du mouvement de chaque pixel (Fleet And Weiss (2005)).

En premier les pixels doivent respecter la cohérence spatiale, c'est-à-dire qu'un pixel dans la première image doit être similaire au pixel correspondant à sa position déplacée selon le mouvement trouvé dans l'image suivante $C_{spatiale} = (X_{t,i,j} - Y_{t+1,i+\delta i,j+\delta j})^2$ où **X, Y** représente les caractéristiques d'un pixel, **i et j** la position du pixel et δ le mouvement trouvé. Deuxièmement, les mouvements trouvés pour chaque pixel doivent se comparer aux mouvements trouvés pour les pixels voisins similaires. Cela revient à dire que les objets dans une image n'ont pas trop de variations de couleur avec le temps et que généralement ils se déplacent d'un mouvement similaire pour tous les pixels le constituant. La plupart des méthodes de flux optique se basent

sur le calcul des gradients pour caractériser les pixels et les retrouver dans l'image suivante (Fleet And Weiss (2005)).

Cependant les calculs simples de gradients se frappent à une limitation, c'est-à-dire que le calcul simple de gradient assume que l'intensité des pixels d'une image à l'autre a très peu varié. Ceci pourrait arriver en présence d'une scène où toutes les surfaces des objets ne présentent aucune spécularité ce qui n'arrive pas souvent pour une scène du monde réel. C'est pourquoi il faut souvent plus qu'un simple gradient. Les problèmes liés à cette situation sont accentués quand ce n'est pas le déplacement des objets dans la scène qui provoque ces changements d'illumination, mais bien des changements provenant de la source d'éclairage elle-même.

1.2.2.1 Visualisation et comparaison du flux optique

Il existe deux méthodes dominantes pour visualiser le flux optique soit la représentation à l'aide de flèches ou l'encodage du mouvement avec des couleurs. La visualisation par flèches est plus facile à interpréter visuellement, mais nécessite souvent un sous-échantillonnage du flux réel puisque sinon les flèches seraient superposées l'une sur l'autre ce qui rendrait difficile la lecture. Ce problème n'est pas présent pour la visualisation par encodage de couleur, mais il faut toujours se rapporter à une carte de couleur représentant le mouvement pour faire le lien entre les couleurs et le mouvement. Ces deux méthodes sont relativement inutiles quand vient le temps de comparer le flux optique d'une image à l'autre cependant, par exemple en présence d'une image de référence pour quantifier la performance d'une méthode. Pour les différences entre deux flux optiques, on utilise fréquemment deux métriques : l'erreur de distance moyenne *Average endpoint error* (AEE) et l'erreur angulaire moyenne *Average angular error* (AAE) qui représentent toutes les deux une mesure moyenne de différence. L'**AEE** est calculée comme la moyenne de la distance absolue entre le point final défini comme la position de départ additionnée du vecteur de mouvement calculé et l'**AAE** comme la moyenne de la différence entre l'angle des vecteurs de mouvements à comparer (Baker *et al.* (2011)).

1.2.2.2 Techniques d'amélioration du flux optique

En partant de la technique de base de calcul du flux optique avec la différence des gradients, plusieurs améliorations sont proposées pour solutionner les problèmes fréquemment rencontrés par celle-ci. Par exemple, cette technique, tout comme celle de la correspondance de blocs emploie une fenêtre de recherche pour ne pas avoir à comparer tous les pixels dans l'image. Ceci introduit la limitation potentielle qu'un mouvement trop important pourrait amener un pixel à sortir de cette zone de recherche et il deviendrait donc introuvable lors de la comparaison. Cette situation amène plusieurs problèmes puisque généralement le calcul du flux optique va prendre le pixel étant le plus similaire, si le pixel correspondant est introuvable, il pourrait arriver que le second pixel le plus similaire soit complètement dans la direction opposée au mouvement réel ce qui entraîne des erreurs importantes. Cette situation est généralement résolue en appliquant un principe de simplification qui sert aussi à régler d'autres problèmes et qui est similaire à une technique retrouvée en BM, la création de pyramides d'images de la plus grossière à la plus fine Black And Anandan (1996).

Le problème de changements d'illumination survient lors de situations qui brisent la prémisse du flux optique qui est que l'illumination des pixels changera peu d'une image à l'autre. Le contre-exemple parfait à cette prémisse serait de faire se déplacer une source d'éclairage autour d'une scène statique. Les zones spéculaires présentes sur les objets vont se déplacer et la couleur des objets va changer aussi conséquemment à la variation d'illumination, ce qui pourrait faire croire à un mouvement qui n'en est pas réellement. Ceci pose problème lors de la recherche du mouvement dans une image puisqu'une zone plus éclairée pourrait avoir l'air de se déplacer dans l'image alors que l'objet est en fait immobile. En général, pour solutionner ce problème, des contraintes sont ajoutées au modèle mathématique utilisé pour l'estimation du flux optique. Entre autres, on retrouve une contrainte de régularisation spatiale qui vient contrebalancer les changements d'illuminations dans une mesure d'erreur d'optimisation (Simoncelli *et al.* (1991)).

D'autres améliorations tournent autour de ce qui décrit un pixel et de ce qui sert à le différencier de ses voisins. Par exemple, on retrouve certaines publications qui utilisent un espace de couleur différent qui permet de séparer la composante d'intensité de celle de la couleur comme *HSV* Mileva *et al.* (2007) ce qui permet de diminuer l'importance des variations d'illumination dans les résultats obtenus. Une autre méthode qui permet d'augmenter la robustesse aux changements d'illumination est de prendre un descripteur de texture pour représenter un pixel ou un descripteur représentant une zone autour du pixel. Par exemple, en calculant une moyenne d'intensité d'une zone et en la comparant avec la variation d'intensité générale dans cette région de l'image on arrive à ignorer partiellement les effets de la variation d'illumination.

Finalement, la technique qui semble obtenir les meilleurs résultats sur l'ensemble d'images de tests utilise, avant d'effectuer l'estimation du flux optique, une segmentation par le mouvement. Cela permet d'avoir un très bon point de départ du flux optique et permet de minimiser l'image de la limitation d'ampleur de mouvement créée par l'utilisation d'une fenêtre de recherche. En ayant déjà une bonne idée des contours des objets, cela permet aussi d'être plus résistant au bruit et aux changements d'illumination puisque normalement on peut s'attendre à ce qu'un objet se déplaçant dans une image conserve plus ou moins sa forme originale (Chen *et al.* (2013a)).

1.2.3 SIFT

Un algorithme très souvent utilisé en traitement d'images et particulièrement en recherche de stéréo correspondance est l'algorithme **SIFT** (*Scale Invariant Feature Transform*). La méthode, publiée par David Lowe (Lowe (1999)), permet d'extraire les caractéristiques intéressantes d'un objet. Ces caractéristiques sont, comme le nom l'indique, invariantes à l'échelle et facilement discernables malgré le bruit, les changements d'illumination et la rotation de l'image. La méthode est basée sur le descripteur SIFT qui est un vecteur de caractéristiques, contenant 128 valeurs, décrivant un point dans une image et qui est construit en utilisant un processus de *différence des gradients*. Ces gradients sont le résultat de l'application de convolutions de gaussiennes successives sur l'image ce qui, en pratique, permet d'éliminer les petits détails

qui ne seraient pas utiles pour reconnaître l'objet et qui permet de conserver les points significatifs. Les points d'intérêts ressortis par la méthode peuvent ensuite être abandonnés suivant une valeur de seuil sur leur importance. En effet, conserver plus de points de correspondance n'entraîne pas forcément un gain au niveau de la recherche et peut parfois causer des erreurs ou complexifier le traitement.

L'algorithme SIFT a premièrement été proposé pour permettre de retrouver des objets provenant d'une image dans une autre image. Pour rechercher la correspondance entre deux images, il suffit de calculer la différence euclidienne entre les descripteurs SIFT calculés sur la première image et ceux de la deuxième image. Comme les descripteurs sont invariants aux changements de taille et très robustes pour ce qui est du bruit et de la rotation, il devient possible de trouver une correspondance entre deux objets qui ont été pris en photo à deux moments différents et à des angles différents. Cette caractéristique peut devenir très utile pour l'estimation du mouvement puisqu'au final, ce que l'on recherche, c'est la correspondance de pixels entre deux images pour arriver à calculer un mouvement.

CHAPITRE 2

MÉTHODE SEMI-AUTOMATIQUE DE GÉNÉRATION DE CARTES DE PROFONDEUR

2.1 Méthode semi-automatique versus automatique

La conclusion tirée de l'analyse de la littérature sur le sujet est qu'il n'existe pas encore de méthode parfaite pour estimer la profondeur. Chaque méthode présente des cas de bon fonctionnement, mais aussi des cas où la méthode donne de mauvais résultats. En effet, pour arriver à de bons résultats avec une méthode automatique, il faut généralement choisir d'avance un indice visuel à utiliser pour aller estimer la profondeur et, comme expliqué, il devient de plus en plus complexe de rajouter plus d'un ou de deux indices de profondeur dans une méthode. Il pourrait être envisageable de précéder le choix de la méthode par une phase d'analyse des caractéristiques et du contenu de la scène pour effectuer un choix sur la méthode à utiliser.

Il existe aussi une autre solution, celle d'utiliser les capacités déjà très puissantes du système visuel humain pour trouver la profondeur. Cela se présente sous la forme d'une méthode semi-automatique d'estimation de la profondeur. Il est possible de demander à un utilisateur d'apporter des annotations de profondeur sur une image pour résoudre le problème de l'estimation. Cette solution a le grand avantage de ne pas dépendre du contenu de la scène car l'information de profondeur provient d'une personne au lieu d'être extraite de l'image elle-même. Puisque de demander à une personne d'annoter toute une image ou même toutes les images d'une séquence serait un processus très long et coûteux, le but d'une méthode semi-automatique est de minimiser l'apport de l'utilisateur tout en lui offrant de garder le contrôle sur le résultat final.

2.2 Méthode proposée

Pour la méthode présentée dans ce mémoire, on demande à une personne d'annoter, sous forme de traits de couleur une ou deux images dans toute la séquence. Ces images s'appelleront des images clés et les annotations nécessaires ne sont que quelques traits sur les différents objets

présents dans l'image. Plus l'utilisateur décide d'être précis dans son annotation, par exemple en utilisant un grand nombre de niveaux différents de profondeur, plus le résultat final sera précis. Une fois les annotations effectuées sur l'image clé, l'algorithme se chargera de propager l'information tout d'abord dans toute l'image puis dans toute la séquence d'images incluse entre les images clés.

2.2.1 Processus d'annotations

En partant de contenu vidéo 2D, l'utilisateur pourra découper le tout en séquence ayant un point de vue semblable et choisir une ou plusieurs images clés au travers de cette séquence. Il annotera ensuite ces images clés sous forme de traits de pinceau d'un ton de gris représentant la profondeur. Il peut annoter jusqu'à 255 niveaux de profondeur dans l'image et il peut décider de lier deux objets différents qui seraient à la même profondeur en plaçant un trait du même ton de gris sur les deux objets. L'utilisateur peut ne mettre qu'un seul point de gris par objet s'il le veut.

2.2.2 Modèle proposé

Le problème est très similaire à un problème de segmentation d'image avec annotations. La solution revient à attribuer une classe de profondeur C à chaque pixel i d'une image I_t. Pour atteindre cette solution, chaque image est représentée sous la forme d'un Champ aléatoire de Markov (*Markov Random Field*) (MRF) qui est un graphe pondéré non orienté. Chaque pixel i de I_t est considéré comme un nœud du graphe et chaque nœud a un ensemble de $N(i)$ voisins. Par exemple, pour un voisinage de taille 8, $N(i)$ contiendrait les 8 pixels les plus proches de i : en haut, en bas, sur les côtés et les diagonales. Pour chaque lien entre deux pixels de l'image i et j on attribue un poids aux arêtes du graphe. Pour représenter ce poids, on utilise deux matrices **W** qui contient la similarité intra-image et **U** la similarité inter-image pour chaque combinaison possible de pixels. Ces matrices sont creuses de par leur définition puisque seulement certaines combinaisons sont possibles, définies par le voisinage.

$$w_{ij} = \begin{cases} exp(-\sigma \cdot ||I_t(i) - I_t(j)||^2) & j \in N(i) \\ 0 & sinon \end{cases} \tag{2.1}$$

Dans le cas où j n'est pas dans le voisinage défini autour de i, ce poids est égal à 0. La valeur de σ peut être modifiée par l'utilisateur pour changer les résultats obtenus, ce paramètre contrôle la diffusion entre les différents pixels. Dans cette expression, $I_t(i)$ représente un vecteur de caractéristiques qui s'appliquent pour le pixel i appartenant à l'image I au temps t. Par exemple, si on fonctionne sous le modèle de couleur RGB, on pourrait avoir comme vecteur de caractéristiques les composantes de couleur du pixel : $I_t(i) = [R;G;B]$.

Considérant une séquence d'image, les pixels vont se déplacer d'une image à l'autre selon les mouvements des objets de la scène. Pour pouvoir propager l'information d'une image I_t vers une image I_{t+1}, il faut tenir compte du mouvement des pixels. Dans le modèle proposé, un ensemble est défini qui représente les déplacements possibles de chaque pixel entre deux images. Cet ensemble est défini selon un noyau de recherche (ou kernel) K. Chaque élément k de ce noyau correspond à un déplacement discret du pixel. L'utilisateur peut contrôler les dimensions de ce noyau. Augmenter la taille du noyau a un impact quadratique sur la complexité des calculs effectués par l'algorithme. La propagation de l'information entre deux images suit la même logique que la propagation intra-image. Elle est représentée par la variable u_{ij} qui représente la similarité entre les deux pixels i et j.

$$u_{ijk} = \sum_{j=1}^{N} d_{ijk} \cdot exp(-\sigma \cdot ||I_t(i) - I_{t+1}(j)||^2) \tag{2.2}$$

Dans cette équation d_{ijk} est égale à 1 si le pixel j dans l'image I_{t+1} correspond au déplacement k à partir du pixel i. Sinon $d_{ijk} = 0$. Ces matrices $\mathbf{W}$ et chaque élément de U_k où $k = 1..K$ (qui est) représentent les poids de chaque arête dans le graphe de relation entre les images de la séquence.

2.2.3 Modèle d'optimisation

Pour résoudre le problème, nous devons assigner à chaque pixel une classe de profondeur $x \in C$ ainsi qu'un déplacement $y_k \in$ K. Le problème est résolu par optimisation en suivant l'équation suivante :

$$minE(x,y) = \sum_{t=1}^{T} E_{WFP}(x_t) + \alpha \sum_{t=1}^{T-1} E_{WFD}(y_t) + \beta \sum_{t=1}^{T-1} E_{AFD}(y_t) + \gamma \sum_{t=1}^{T-1} E_{AFPD}(x_t, x_{t+1}, y_t) \tag{2.3}$$

avec $\mathbf{x_t}$ représentant la profondeur, et $\mathbf{y_t}$ représentant les déplacements en fonction du temps (avec les indices i et j, représentant la position dans l'image omis pour clarté).

Avec les contraintes :

$$\sum_{k=1}^{K} y_{tik} = 1, i \in 1,\ldots,N, t \in 1,\ldots,T-1 \tag{2.4}$$

$$x_{ti} = c, \forall (i,c) \in S_t, t = 1,\ldots,T \tag{2.5}$$

L'équation d'énergie précédente est composée de quatre différents termes qui servent à contrôler les résultats obtenus. L'importance de chacun de ces termes est contrôlée par α, β, γ qui servent à pondérer chaque portion de l'équation d'énergie et qui permettent plus de contrôle sur les résultats obtenus. Le premier terme représente la cohérence de profondeur intra-image. Ce terme pénalise la différence de classe de profondeurs attribuée à deux pixels, pondérée par la similarité entre ces pixels tout en restant à l'intérieur du noyau de recherche. Ce terme est basé sur la prémisse qu'à l'intérieur des objets, il n'y a que des petites variations de profondeur. Dans ce terme, $w_{i,j}$ est la similarité entre les pixels i et j et $x_{t,i|j}$ représente la classe de profondeur attribuée au pixel i ou j au temps t. Le terme est défini comme suit (le t a été abandonné des équations suivantes pour clarté) :

$$E_{WFP}(x_t) = \frac{1}{2}\sum_{i,j=1}^{N} w_{ij} \cdot (x_i - x_j)^2 \tag{2.6}$$

Le second terme, semblable au premier, représente la cohérence de déplacement intra-image. C'est-à-dire qu'il pénalise deux pixels ayant une forte connectivité qui auraient différents déplacements attribués. Dans cette équation, $y_{t,i|j,k}$ représente le déplacement attribué au pixel i ou j au temps t et ce déplacement est représenté par une valeur contenue dans le noyau K en tant que k. Ce terme assume que les objets bougent généralement dans la même direction et est défini comme suit :

$$E_{WFD}(y_t) = \frac{1}{2}\sum_{i,j=1}^{N}\sum_{k=1}^{K} w_{ij} \cdot (y_{ik} - y_{jk})^2 \tag{2.7}$$

Le troisième terme représente la cohérence de déplacement inter-image. Il pénalise l'assignation d'un mouvement à un pixel qui le ferait correspondre à un pixel qui est différent dans l'image suivante. Dans cette équation u_{ik} représente la similarité entre le pixel i dans l'image au temps t et le pixel correspondant à la position du pixel i déplacée par le mouvement k dans l'image au temps $t+1$. Il est défini comme suit :

$$E_{AFD}(y_t) = \sum_{i=1}^{N}\sum_{k=1}^{K} u_{ik} \cdot (1 - y_{ik}) \tag{2.8}$$

Le dernier terme représente la cohérence de profondeur inter-image. Ce terme pénalise l'assignation d'un déplacement k à un pixel i qui le ferait correspondre à un pixel j dans l'image $t+1$ qui a une grande différence de profondeur attribuée x. Il est défini comme suit :

$$E_{AFPD}(x_t, x_{t+1}, y_t) = \sum_{i,j=1}^{N}\sum_{k=1}^{K} d_{ijk} \cdot y_{ik} \cdot (x_{ti} - x_{t+1,j})^2 \tag{2.9}$$

L'équation impose aussi deux contraintes, la première s'assure qu'un pixel n'a qu'un seul déplacement attribué lors de la phase d'optimisation et la seconde contrainte est formée par les annotations effectuées par l'utilisateur. Si un pixel a été annoté, sa profondeur est attribuée avec une probabilité de 100% et ce pixel ne sera pas modifié par le processus d'optimisation. Cette seconde contrainte a un effet double car en plus d'ignorer ce pixel pour l'optimisation, la probabilité de 100% aura un grand effet sur la propagation de la confiance attribuée à la profondeur.

Puisque les variables que nous cherchons x_{ti} et y_{tik} sont interdépendantes par le terme E_{AFPD}, ce problème devient très difficile à optimiser. C'est pourquoi le processus de minimisation est divisé en deux phases qui vont se répéter en alternance jusqu'à atteindre une convergence ou un nombre maximum d'itérations. Dans la première phase, on optimise la variable x_{ti} en comptant y_{tik} comme étant fixe puis l'inverse est exécuté durant la seconde phase.

2.2.4 Calcul des classes de profondeur

Durant cette étape, il faut trouver la profondeur associée à chaque pixel en considérant les déplacements comme fixes. À ce niveau, seuls les termes E_{WFP}, E_{AFPD} ainsi que la contrainte concernant les annotations sont considérés puisque l'on assume les autres constantes. Pour résoudre le problème, on exprime alors les termes concernés sous forme matricielle. Prenons $x_t \in \{0,1\}^N$ soit un vecteur des classes de profondeur pour une image I_t et L_t la matrice laplacienne $N \times N$ de I_t telle que :

$$[L_t]_{ij} = \begin{cases} \sum_{j=1}^{N} w_{tij} & i = j \\ -w_{tij} & sinon \end{cases} \tag{2.10}$$

Et aussi la matrice A_t qui est aussi une matrice $N \times N$ qui ajoute un lien dynamique entre un pixel et le pixel correspondant à son déplacement dans l'image suivante. La matrice est définie comme suit :

$$[A_t]_{ij} = \sum_{k=1}^{K} d_{ijk} \cdot y_{tik} \tag{2.11}$$

On définit aussi les matrices C_t et D_t qui sont les matrices diagonales $N \times N$ qui représentent respectivement la somme en i de A_t et la somme en j de A_t. On peut donc exprimer le problème comme la minimisation de l'équation d'énergie suivante :

$$\begin{aligned} E(x) &= \sum_{t=1}^{T} x_t^T L_t x_t + \gamma \sum_{t=1}^{T-1} \left(x_t^T D_t x_t - 2 x_t^T A_t x_{t+1} + x_{t+1}^T C_t x_{t+1} \right) \\ &= \sum_{t=1}^{T} x_t^T \underbrace{\left(L_t + \delta_{t<T}(\gamma D_t) + \delta_{t>1}(\gamma C_{t-1}) \right)}_{Q_t} x_t - \delta_{t<T}(2\gamma x_t^T A_t x_{t+1}) \end{aligned} \tag{2.12}$$

Dans cette équation, $\delta_{condition}(x)$ représente la fonction de Dirac, si la condition est vraie, $\delta_{condition}(x) = x$ et si la condition est fausse, $\delta_{condition}(x) = 0$. La solution à cette équation peut être trouvée globalement, pour toutes les images de la séquence vidéo, mais en raison d'un grand nombre potentiel d'images, une solution itérative, une image à la fois est préférée.

2.2.4.1 Première étape : Résolution de la profondeur des pixels

Pour résoudre l'équation précédente, une solution itérative est proposée. Cette solution va résoudre l'équation, une image à la fois, en assumant que les autres images de la séquence ont déjà leur profondeur résolue. La solution revient donc à résoudre cette équation qui est relative au temps dans la séquence :

$$E(x_t) = x_t^T Q_t x_t - 2\gamma x_t^T \underbrace{\left(\delta_{t<T}(A_t x_{t+1}) + \delta_{t>1}(A_{t-1}^T x_{t-1}) \right)}_{r_t} \tag{2.13}$$

Pour tenir compte des annotations de l'utilisateur, on réarrange l'équation selon les pixels annotés et non annotés :

$$E(x_{U_t}) = \begin{bmatrix} x_{Ut}^T & x_{Lt}^T \end{bmatrix} \begin{bmatrix} Q_{UUt} & Q_{ULt} \\ Q_{ULt}^T & Q_{LLt} \end{bmatrix} \begin{bmatrix} x_{Ut} \\ x_{Lt} \end{bmatrix} - 2\gamma \begin{bmatrix} x_{Ut}^T & x_{Lt}^T \end{bmatrix} \begin{bmatrix} r_{Ut} \\ r_{Lt} \end{bmatrix} \tag{2.14}$$

En cherchant les valeurs où la dérivée est nulle et en relaxant les contraintes sur les valeurs entières on trouve la fonction suivante :

$$x_{Ut} = -Q_{UUt}^{-1}(Q_{ULt}x_{Lt} - \gamma r_{Ut}) \tag{2.15}$$

En prenant les images annotées par l'utilisateur comme des images clés dans la séquence, les meilleurs résultats pourront être obtenus en diffusant l'information en partant des images clés vers le reste des images. Par exemple, on trouve les classes de profondeur de l'image annotée au temps $t = 0$ et ensuite en résolvant les images à distance de 1 de l'image clé, puis 2 etc. jusqu'à ce que toutes les images aient été résolues.

2.2.4.2 Deuxième étape : Résolution des déplacements de pixels

Une fois la profondeur des pixels calculée, on peut maintenant fixer cette partie du problème et calculer maintenant les déplacements des pixels. On définit le vecteur y_{tk} dont le *i-ème* élément correspond à y_{tik} ainsi que le vecteur b_{tk} où le *i-ème* élément est

$$[b_{tk}]_i = -\beta u_{tik} + \gamma \sum_{j=1}^{N} d_{ijk} \left(x_{ti} - x_{t+1,j}\right)^2 \tag{2.16}$$

L'application de cette technique en ne considérant que deux classes de profondeur (une pour l'avant-plan et une pour l'arrière-plan) est similaire a un problème plus classique de segmentation par graphe. Une des classes de profondeur serait 1 et l'autre 0. Dans ce cas, la formulation de cette équation est équivalente à 2.16. Le second sous-problème peut être exprimé comme suit :

$$
\begin{aligned}
minE(y) &= \sum_{t=1}^{T-1} \sum_{k=1}^{K} \alpha y_{tk}^T L_t y_{tk} + b_{tk}^T y_{tk} \\
&\text{avec les contraintes} \\
&\sum_{k=1}^{K} y_{tk} = 1, t = 1, \ldots, T-1
\end{aligned}
\tag{2.17}
$$

Pour trouver la solution, il faut grouper tous les déplacements pour chaque image

$$
y_t^T = \begin{bmatrix} y_{t1}^T & \ldots & y_{tK}^T \end{bmatrix} \tag{2.18}
$$

On doit aussi définir les matrices en bloc suivantes :

$$
S_t = \begin{bmatrix} \underbrace{\begin{matrix} \alpha L_t & \ldots & 0 \\ 0 & \ldots & 0 \\ 0 & \ldots & \alpha L_t \end{matrix}}_{\text{K fois}} \end{bmatrix} \tag{2.19}
$$

et aussi

$$
G = \begin{bmatrix} \underbrace{\begin{matrix} I_{N\times N} & \ldots & I_{N\times N} \end{matrix}}_{\text{K fois}} \end{bmatrix} \tag{2.20}
$$

et le vecteur bloc suivant :

$$
b_t^T = \begin{bmatrix} b_{t1}^T & \ldots & b_{tK}^T \end{bmatrix} \tag{2.21}
$$

On peut exprimer la fonction d'énergie avec contraintes en forme matricielle comme ceci :

$$E(y) = \sum_{t=1}^{T-1} y_t^T S_t y_t + b_t^T y_t$$
$$\text{avec les contraintes} \tag{2.22}$$
$$Gy_t = 1, t = 1, \ldots, T-1$$

Et, puisque les déplacements des pixels d'une image sont indépendants des déplacements de pixels des autres images dans la fonction d'énergie et dans les contraintes, on peut optimiser chaque y_t de façon individuelle pour t=1, …, T-1

$$E_t(y_t) = y_t^T S_t y_t + b_t^T y_t$$
$$\text{avec les contraintes} \tag{2.23}$$
$$Gy_t = 1$$

Si on remplace les contraintes entières sur les variables y_{tik} avec des contraintes de valeurs minimales $t_{tik} \geq 0$ ce problème correspond à un problème d'optimisation convexe quadratique dont le minimum global peut être obtenu avec une méthode comme la Méthode des points intérieurs *Interior Point Method*.

2.2.5 Algorithme global

Une représentation de l'algorithme général de la méthode est décrite dans la figure suivante.

Algorithm 1: Génération semi-automatique de cartes de profondeur

Input: Séquence d'images $\mathbf{I_{1..T}}$;
Input: Les annotations de l'utilisateur $\mathbf{S_{1..T}}$;
Input: Le noyau de recherche $\mathbf{K}$;
Input: Le paramètre qui contrôle la diffusion σ ;
Input: Les paramètres de contrôle α, β, γ ;
Output: La profondeur des pixels $\mathbf{x_{ti}}$;
Output: Le déplacement des pixels $\mathbf{y_{tik}}$;

— Initialization —

$$\mathbf{y_{tik}} = \frac{\mathbf{1}}{\mathbf{K}} \forall \mathbf{t}, \forall \mathbf{i}, \forall \mathbf{k};$$

— Boucle principale —
repeat
  % Calcul de la profondeur
  foreach *image I_t en s'éloignant de l'image clé* **do**
    Ajuster la profondeur des pixels de l'image $\mathbf{x_{ti}}$ (éq. 2.15)

  % Calcul du mouvement
  foreach *image I_t dans la séquence d'images* **do**
    Ajuster le déplacement des pixels $\mathbf{y_{tik}}$ (éq. 2.23)
until *Convergeance ou un nombre maximal d'opérations atteint*;
return $\{\mathbf{x_{ti}}, \mathbf{y_{tik}}\}$;

Figure 2.1 Méthode itérative pour calculer la profondeur des pixels et leurs déplacements

CHAPITRE 3

ESTIMATION DE MOUVEMENT

La difficulté principale de la méthode présentée au chapitre précédent est la propagation adéquate de l'information annotée sur l'image clé vers les autres images qui ne sont pas annotées. Avec cette information, le transfert de données tout au long d'une séquence d'image est possible de façon très précise, car chaque pixel peut être relié à une trajectoire dans toute la séquence. Donc, une fois l'image clé "segmentée", la profondeur associée à chaque pixel dans l'image clé peut être propagée sur la séquence. Dans le cas où cette profondeur changerait durant la séquence, elle serait interpolée à partir d'une image clé au début de la séquence et une autre à la fin de la séquence. C'est pourquoi le problème de l'estimation de mouvement se présente comme une étape à résoudre pour arriver à de bons résultats à la problématique présentée précédemment.

Pour obtenir l'information nécessaire, il faut arriver à déterminer le déplacement de chaque pixel entre deux images consécutives dans une séquence. La méthode employée pour y arriver présente des similarités avec la méthode de propagation pour la génération de cartes de profondeur dans le sens où elle se base sur un processus qui utilise la similarité entre les pixels pour propager de l'information.

3.1 Random Walker

La méthode proposée utilise un processus appelé un "Random walker" qui est originalement une technique pour la segmentation d'image, mais qui est utilisée ici pour l'estimation de mouvement. Dans ce processus, l'image est considérée comme un graphe dont chaque sommet x_i représente les pixels de l'image I et ayant des arêtes pondérées entre les pixels selon la similarité. La similarité entre les pixels est généralement basée sur la somme du carré de la différence entre les descripteurs de chaque pixel. Chaque pixel dans l'image est représenté selon un descripteur qui prend la forme d'un vecteur de caractéristiques x_i. La similarité entre deux pixels i et j et le processus générique sont définis comme ceci

$$w_{ij} = exp(-\gamma \cdot ||x_i - x_j||^2) \tag{3.1}$$

où le terme γ sert à contrôler le poids de chaque élément dans le vecteur descripteur de chaque pixel. Les vecteurs servants à représenter chaque pixel de l'image peuvent fortement influencer l'efficacité des résultats obtenus et leur composition sera discutée plus en détail dans les sections présentant les résultats ainsi que la discussion sur la méthode qui expliquera les choix effectués pour la constitution de ces descripteurs.

Algorithm 2: Random Walker

— Initialization —
% Annotation de l'image par l'utilisateur
— Processus principal —
foreach *pixel i non annoté dans l'image I* **do**
    foreach *classe d'annotation utilisée c* **do**
        Calculer la probabilité d'atteindre *c* en partant de *i*
    Assigner la classe *c* ayant la plus haute probabilité au pixel *i*

Figure 3.1 Définition générale d'un Random Walker

Une analogie qui représente bien le principe du Random Walker est celle d'un circuit électrique ayant des nœuds et des résistances entre les nœuds proportionnelles à la similarité entre les pixels. En choisissant d'un point de départ sur le circuit, on peut calculer le point d'arrivée le plus probable en choisissant celui ayant la moindre résistance et on attribue la classe du point d'arrivée le plus probable au point de départ.

3.2 Méthode proposée

Étant donné une image I_t et les images consécutives dans une séquence I_{t+n}, chaque pixel d'une image i est associé à un vecteur descripteur x_i. Le problème revient à attribuer à chaque pixel i de l'image I_t un déplacement $y_i \in K$. K représente un noyau de recherche de similarité dans l'image suivante I_{t+1} ayant une taille définie par l'utilisateur. L'assignation de vecteurs de mouvements à chaque pixel est basée sur la similarité entre les pixels inter-images, mais chaque pixel est aussi influencé par ses voisins similaires selon le principe du *Random Walker*. La méthode proposée tire ses racines dans les principes d'optimisation linéaire puisqu'un modèle probabiliste est utilisé qui permet ultimement de sélectionner le déplacement le plus probable pour chaque pixel en fonction de plusieurs facteurs considérés dans le modèle. La méthode est paramétrable, ce qui permet un contrôle plus fin au niveau des résultats obtenus, mais ce qui oblige aussi de trouver la bonne valeur pour chaque paramètre pour obtenir le résultat optimal, ce qui signifie parfois qu'il faut employer un processus itératif.

Il est à noter que les meilleurs résultats obtenus avec la méthode l'ont été en utilisant l'espace de couleur Lab. Cet espace de couleur comprend trois composantes comme la couleur RVB, mais la différence principale est que la composante L représente la luminosité tandis que les composantes A et B représentent des coordonnées dans un espace de couleurs non linéaire. Cette transformation est importante, car cela permet de séparer l'intensité d'un pixel de sa couleur ce qui permet de plus facilement contrôler l'importance de ces deux caractéristiques dans le descripteur de pixels utilisé.

Le principe général de la méthode est décrit par l'algorithme suivant :

Algorithm 3: Estimation du mouvement entre deux images

Input: Deux images I_1 et I_2 ;
Input: Taille du noyau intra (K_w) et inter-images(K_u) ;
Input: Les paramètres γ qui contrôlent l'importance de chaque partie du descripteur ;
Input: Le paramètre de contrôle de la diffusion β ;
Output: La probabilité de déplacement de chaque pixel M_{NxK} ;

— Initialization —
% Calcul des voisins selon les fenêtres de recherche K_w et K_u
— Calcul de la similarité entre voisins —
foreach *ligne dans la liste de voisinage* **do**
    % Calcul de W, la matrice de similarité entre un pixel et ses voisins dans la même image.
    % Calcul de U, la matrice de similarité entre un pixel et ses voisins dans la fenêtre de recherche dans l'image suivante
% On normalise ensuite W et U en P et Q respectivement en divisant chaque valeur par la valeur totale par ligne

% On calcule ensuite la probabilité de chaque mouvement avec l'opération suivante qui est l'équivalent de résoudre un système linéaire :

$$Motion = (I - (I - \beta) \cdot P) \backslash (\beta \cdot Q)$$

% On peut ensuite prendre le maximum par ligne du résultat pour avoir le mouvement le plus probable.
return $\max(Motion, 2)$;

Figure 3.2 Algorithme général d'estimation de mouvement

3.2.1 Calcul de la similarité intra-image

En premier lieu, la similarité intra-image est calculée, basée sur la somme des différences absolues des vecteurs descripteurs de chaque pixel compris dans le voisinage défini par l'utilisateur. La matrice contenant les similarités est notée W :

$$w_{ij} = \begin{cases} exp(-\gamma_w \cdot ||x_i - x_j||) & j \in K_w \\ 0 & sinon \end{cases} \tag{3.2}$$

Le paramètre γ sert à contrôler le poids de chaque composante dans le vecteur descripteur des pixels. Cela va créer une matrice W qui sera N (nombre de pixels dans l'image) par N mais qui sera creuse car seulement les pixels voisins vont avoir une similarité et la similarité des pixels non voisins sera de 0. Dans une matrice creuse, seulement les valeurs manuellement affectées sont stockées, alors toutes les combinaisons de coordonnées où les pixels ne sont pas voisins ne représentent pas de complexité additionnelle pour les calculs futurs. Donc en réalité, la matrice contient environ $N \cdot K_w$ valeurs, ce qui est très inférieur à la taille d'une matrice N par N.

Pour les calculs suivants, il est nécessaire de normaliser la matrice résultante car la valeur représente une probabilité qui doit totaliser 1. Le résultat normalisé de W est appelé P et il est obtenu en divisant chaque valeur sur une ligne par le total de cette ligne.

3.2.2 Calcul de la similarité inter-images

En deuxième, on calcule la similarité inter-images, c'est-à-dire la similarité entre un pixel de l'image I_t et les pixels dans une fenêtre de recherche dans l'image I_{t+1}. Encore une fois, la similarité est basée sur la somme absolue des différences dans le voisinage défini par l'utilisateur. La matrice contenant les similarités est notée U :

$$u_{ik} = exp(-||\gamma_u \cdot (x_i - x_{i+k,t+1})||) \tag{3.3}$$

Le paramètre γ_u sert à contrôler le poids de chaque élément dans le vecteur descripteur des pixels. Dans le cas de la matrice U, au lieu d'obtenir une matrice $N \times N$, on obtient seulement une matrice NxK, où K est le nombre de pixels inclus dans le voisinage défini par l'utilisateur et cette matrice n'est pas creuse mais pleine, cette matrice sera beaucoup plus lourde pour les calculs suivants mais il est nécessaire que cette matrice soit pleine pour les opérations suivantes.

Pour les calculs suivants, il est nécessaire de normaliser la matrice résultante. Le résultat normalisé de U est appelé Q, comme pour la matrice W, la normalisation est obtenue en divisant

chaque valeur d'une ligne par le total de cette ligne. Cette manipulation va faire que la somme de chaque ligne va être de 1 pour représenter 100% probabilité totale.

3.2.2.1 Scale-Invariant Feature Transform

Durant les expérimentations, une observation qui ressortait était que la confiance obtenue (la plus haute valeur probabiliste obtenue par les calculs pour un pixel) dans certaines zones de l'image était plutôt faible, surtout dans les zones fournissant peu d'indices comme les zones dans le milieu de blocs sans texture ou les zones ayant des textures répétitives. Comme la méthode attribue une probabilité de mouvement selon la similarité des pixels, dans ces cas de figure, la probabilité de mouvement était souvent équivalente entre plusieurs pixels. Il fallait trouver un moyen pour que la méthode puisse arriver à choisir entre toutes les possibilités de déplacement. C'est pour cette raison que l'option d'inclure des correspondances calculées par SIFT a été ajoutée à la méthode. Un descripteur SIFT étant généralement accepté comme une technique fiable pour trouver des correspondances significatives entre deux images, cette méthode s'en sert pour influencer certaines zones de pixels.

Les points significatifs SIFT sont calculés sur chacune des deux images et ensuite chaque point est comparé pour trouver une correspondance selon un certain seuil de similarité. Seuls les points dépassant un fort seuil de similitude sont conservés et sont par la suite filtrés pour conserver seulement les pixels qui donnent une correspondance incluse dans la zone de mouvements possibles définie avec le noyau de recherche. Ceux-ci servent par la suite à remplacer la probabilité de mouvement des pixels calculés par la méthode. En fait, après avoir calculé la matrice U, les pixels où des correspondances SIFT ont été conservées se voient attribuer une similarité de 1 (qui correspond à la plus haute similarité possible) à l'endroit de la correspondance et 0 (la plus faible similarité) pour tous les autres pixels. En termes concrets, cela aura l'effet de mettre la probabilité du mouvement à près de 100% pour ces pixels. Ces pixels influenceront aussi fortement leurs voisins rapprochés puisque la diffusion tient compte de la certitude.

$$u_{ik} = \begin{cases} 1 & \text{Si j et k sont correspondant selon SIFT} \\ 0 & \text{Pour toutes les autres valeurs de k dans K} \end{cases} \tag{3.4}$$

Dans l'image suivante, on a marqué les points qui sont les plus significatifs pour SIFT sur l'image originale. On suit le même processus dans la seconde image et ensuite, pour chaque point SIFT, on essaie de trouver une correspondance dans les points SIFT de l'image suivante. Ensuite, on calcule la différence de position entre les deux points correspondants et on prend cette distance comme étant le vecteur de mouvement de ce pixel

Figure 3.3 Points SIFT de l'image 1

L'image suivante représente la confiance de la méthode d'avoir trouvé le bon déplacement. Cette image a été générée avec le paramètre Beta à une valeur de 1 ce qui signifie qu'aucun pixel n'influence ses voisins et seulement la correspondance trouvée décide de la confiance des mouvements. Les points qui sont totalement blancs dans une zone autrement plus foncée sont les pixels qui ont été trouvés comme étant des correspondances SIFT.

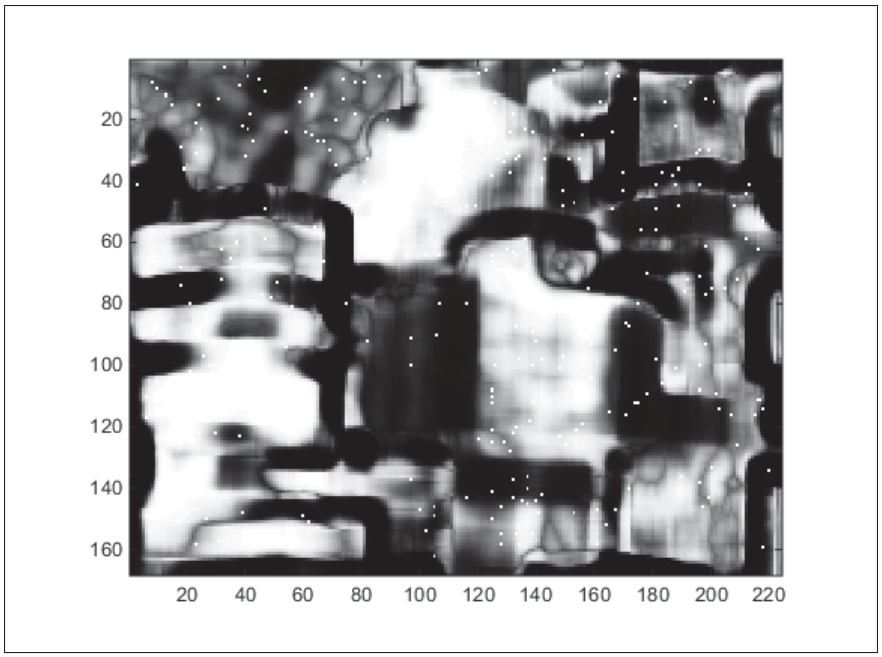

Figure 3.4 Confiance de la méthode avec SIFT

3.2.3 Calcul de la probabilité des déplacements

Avec les similarités intra-image et inter-images calculées, la formulation du problème peut être traduite par le processus du Random Walker, qui peut être représenté par un système d'équations linéaires. La probabilité de chaque mouvement pour chaque pixel est calculée dans la matrice Y selon la formulation suivante :

$$Y = (I - (1-\beta)\cdot P)\backslash(\beta \cdot Q); \tag{3.5}$$

La formule avec l'anti-slash représente une instruction Matlab qui permet de résoudre un système d'équations linéaires (elle représente la méthode d'élimination de Gauss). Elle représente la solution au problème $Ax = B => x = A\backslash B$. Dans cette formulation β est la probabilité d'arrêter de se déplacer avec le random walker, un paramètre qui contrôle la propagation de l'information. Donc, plus le paramètre β est petit entre $[0..1]$ plus l'information de mouvement sera propagée à travers les pixels. Une fois calculée, la matrice Y (N par K) contiendra la probabilité

de chaque pixel de se déplacer à chaque déplacement possible défini par le noyau de recherche dans l'image suivante.

3.3 Superpixels

Dans l'optique d'améliorer la rapidité de la méthode et en espérant aussi régler certains des problèmes rencontrés, un prétraitement a été développé pour séparer l'image en superpixels et utiliser ceux-ci pour trouver le mouvement dans l'image au lieu d'effectuer le calcul pixel par pixel. Les superpixels (SP) sont des zones dans l'image, généralement de petite taille, qui suivent normalement les frontières des objets. La génération de ceux-ci est basée sur la similitude des couleurs tout en suivant un paramètre guide pour déterminer la taille désirée de ces SP, cependant cette taille varie fortement en fonction du paramètre seuil qui décide à quel point le SP tient compte des couleurs similaires. Cette façon de procéder va évidemment dans la même direction que la méthode proposée puisque la matrice W tient déjà compte de la similarité entre les pixels voisins pour la diffusion de l'information.

Le fonctionnement de la méthode reste similaire sauf qu'au lieu de calculer la similarité inter-images pour tous les pixels, elle n'est calculée que pour un certain nombre de pixels choisis au hasard dans chaque superpixel. Pour tenir compte de cette modification, une matrice nommée A est créée qui contient l'information d'appartenance de chaque pixel à son superpixel et la matrice C représente la matrice A mais normalisée. À l'aide de la matrice A on peut savoir quel pixel appartient à quel superpixel et cette appartenance est utilisée pour les calculs puisque l'estimation de mouvements ne se fait plus sur tous les pixels mais seulement sur un certain nombre de pixels par superpixels. La formulation qui sert à résoudre le système d'équations linéaire est modifiée pour tenir compte de ces nouvelles contraintes.

$$Y = (I - (1-\beta)\cdot P) \backslash (\beta \cdot C^T Q); \tag{3.6}$$

En ajoutant la transposée de la matrice C, qui est la matrice A normalisée, cela change l'équation pour que Q ne tienne maintenant compte que des pixels qui ont été choisis au hasard dans chacun des superpixels. Comme le nombre de calculs effectués dans la méthode proposée est proportionnel au nombre de pixels considérés, une technique de simplification qui réduit le nombre de pixels à considérer a un impact significatif sur le temps de calcul.

Par exemple dans l'image suivante, qui a une taille de 640 par 480, si on choisit une taille moyenne de SP de 5 (ce qui signifie de générer en moyenne des SP qui font 5 par 5), cela nous donne précisément 1530 SP et l'on choisit de considérer seulement 15 pixels par SP. Cela implique que la solution va considérer seulement $1530 \times 15 = 7650$ pixels au lieu de $640 \times 480 = 307200$ pixels. Le temps de calcul dépend de la machine utilisée, mais on voit facilement la différence entre le nombre de pixels qu'il aurait fallu calculer normalement versus le nombre de pixels à calculer avec les SP (une diminution de 98%). La figure 3.5 montre l'image originale séparée en SP.

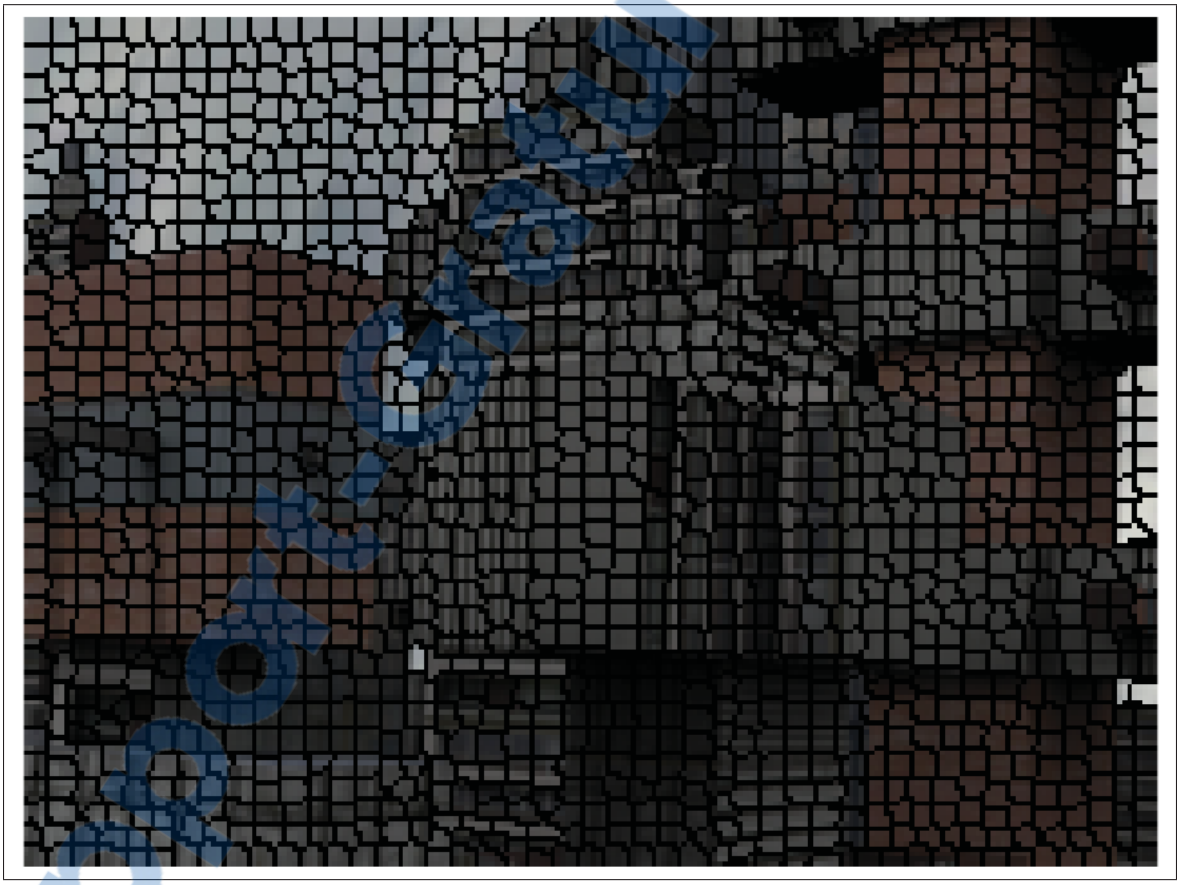

Figure 3.5 Superpixels sur l'image Urban3

CHAPITRE 4

RÉSULTATS OBTENUS - GÉNÉRATION DE CARTES DE PROFONDEUR

Les expérimentations effectuées avec la méthode de génération de cartes de profondeur sont relativement limitées car la méthode s'est rapidement frappée à plusieurs problèmes. La méthode évaluée obtient de bons résultats en ce qui a trait à l'estimation de la profondeur sur une image à partir des annotations d'un utilisateur. Par contre, comme mentionné plus haut, pour arriver à propager l'information à travers le temps à partir des images clés qui ont été annotées, une bonne technique permettant de trouver la correspondance exacte des pixels entre les images est un pilier nécessaire à l'obtention de bons résultats. Puisque la méthode tente de limiter le nombre d'images à annoter dans une séquence, cela signifie souvent d'avoir à propager une première estimation de profondeur sur plusieurs images. Dans le cas où la correspondance entre les pixels ne serait pas bonne, on se retrouverait dans une situation où une erreur serait propagée et amplifiée au travers de la séquence. Ce qui concrètement créerait un effet de *ghosting* ou une trace qui serait décalée par rapport au mouvement réel. Si ce résultat était utilisé par exemple pour de la conversion 2D à 3D cela donnerait un résultat erroné qui serait facilement visible une fois visionné.

C'est ce besoin important d'une technique d'estimation de mouvement relativement rapide qui a justifié les expérimentations plus importantes pour l'estimation de mouvement. Cependant, les résultats obtenus avec des images simplistes sont prometteurs pour la poursuite de la recherche. En pratique les expérimentations ont été effectuées sur des images contenant deux niveaux de profondeur, c'est-à-dire avec un objet en premier plan et un arrière-plan. Un système d'annotation a été mis en place qui permet à l'utilisateur de placer un ou plusieurs points dans l'image en spécifiant la "classe de profondeur" de ce(s) point(s). Cela veut dire qu'en ayant multiples classes de profondeur, l'utilisateur peut spécifier si un objet se trouve plus en avant d'un autre objet en indiquant une classe de profondeur plus ou moins profonde. Il ne suffit que de quelques points par objet pour obtenir une bonne propagation (similaire à de la

segmentation semi-automatique) qui permet d'attribuer la classe de profondeur à tous les pixels de l'objet sans débordement.

4.1 Images générées

Les premiers tests effectués sont sur des ellipses se déplaçant horizontalement noir sur blanc et les images contiennent un niveau élevé de bruit. Le but de ces tests était d'arriver à attribuer une valeur de profondeur à l'ellipse la séparant du fond, mais ce sur la sixième image de la séquence tout en ayant annoté seulement la première image avec un petit nombre de points. La figure 4.1 présente la séquence d'images sur laquelle les premiers tests ont été effectués et la figure 4.2 présente les annotations effectuées par l'utilisateur pour indiquer la profondeur.

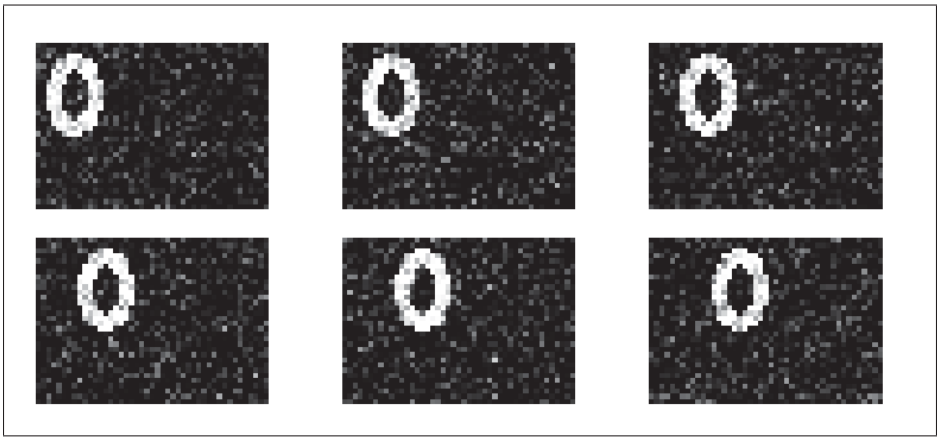

Figure 4.1 Séquence d'images d'ellipses bruitées

Les points bleus représentent la classe de profondeur 0 (par exemple qui pourrait représenter l'avant-plan) et les points rouges représentent la classe 1 qui correspond au fond de l'image. À noter que le centre de l'ellipse, bien qu'étant très différent du blanc de l'ellipse a été annoté comme faisant partie de l'avant-plan pour montrer que cette méthode n'est pas basée sur la segmentation de la couleur de l'image. Les résultats obtenus montrent que la méthode tient bien compte des annotations plus que de la couleur des pixels pour assigner des classes de profondeur. Il faut noter que dans la séquence d'images utilisée, seule la première image a été

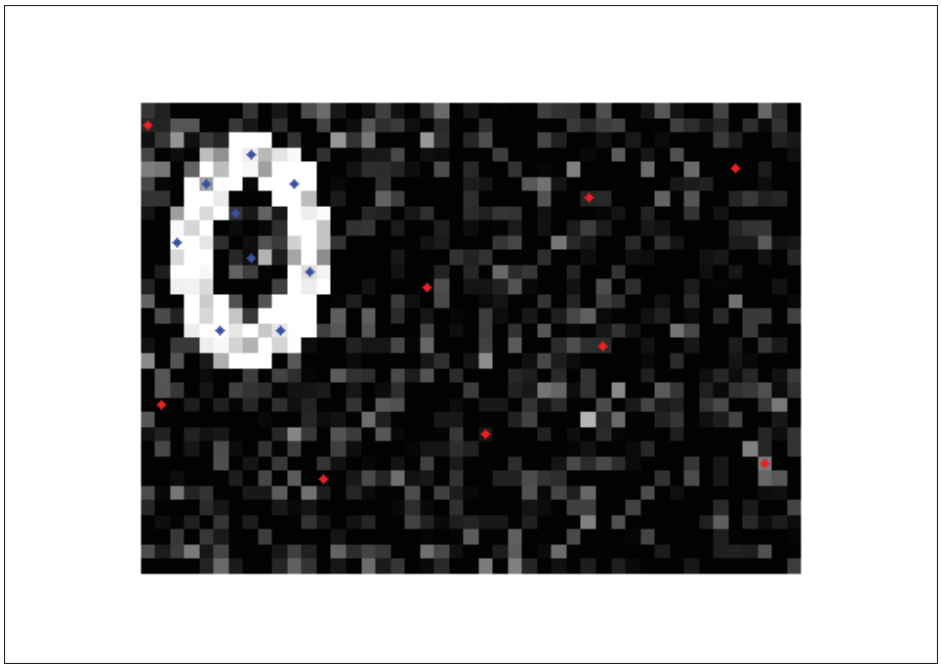

Figure 4.2 Premières images testées (bruitées)

annotée, donc les résultats obtenus sur les autres images ne proviennent que de la propagation de l'information de la méthode.

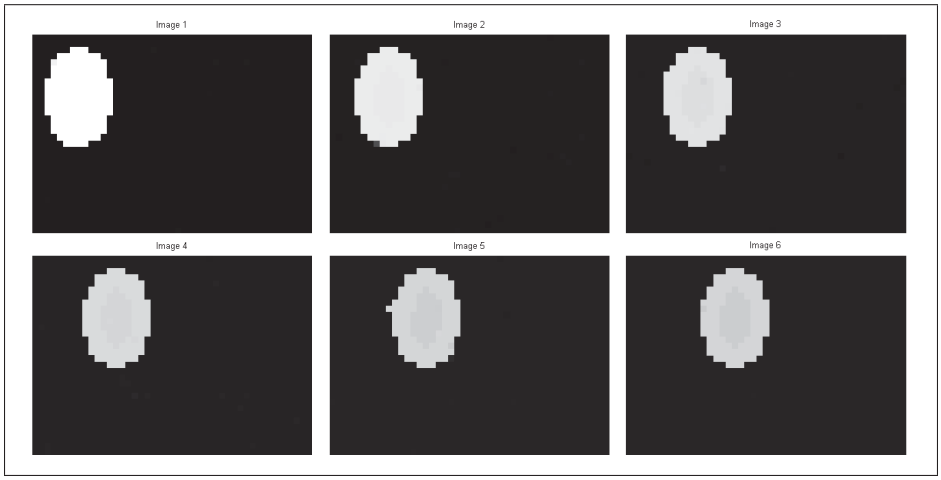

Figure 4.3 Résultats séquence ellipse - L'intensité représente la classe de profondeur attribuée

Puisque la méthode est basée sur un système de probabilité, on observe que le résultat obtenu n'est pas un absolu noir ou blanc mais plutôt que chaque pixel a une teinte de gris qui se rapproche de la valeur qu'il doit avoir au final. Les variations de teintes de gris correspondent à des variations dans la certitude de l'assignation de la profondeur. On peut observer que plus on s'éloigne de l'image clé qui a été annotée moins la certitude est élevée mais que l'on conserve tout de même une différence importante dans l'assignation de la profondeur aux pixels. En appliquant une fonction de séparation, par exemple, à 0.5 on obtient clairement une séparation entre le premier plan et le fond de l'image.

Le test suivant pour la méthode a été de passer à des images couleur, cependant moins bruitées et de varier quelque peu la forme des objets pour estimer la profondeur. Encore une fois, dans le test suivant, le centre de la petite ellipse a été annoté comme faisant partie de l'objet pour s'assurer que la méthode ne segmentait pas seulement la couleur. La figure 4.4 montre les annotations effectuées par l'utilisateur et les figures 4.5 à 4.7 montrent les résultats obtenus par la méthode sur trois itérations. On remarque une amélioration de la certitude de l'assignation de la profondeur au fil des itérations ce qui est le résultat attendu. Les ellipses présentent encore un mouvement de translation horizontal linéaire dans la séquence. Une amélioration a été appliquée sur cette version de la méthode par rapport aux ellipses en noir et blanc, au lieu d'utiliser un noyau (*kernel*) de recherche carré, le noyau utilisé était sous forme de losange. Dans notre cas, utiliser un noyau de recherche de cette forme présente une amélioration car il est possible d'augmenter la distance de recherche de pixels similaires sur l'axe horizontal et vertical sans augmenter le nombre total de pixels comparés.

Les résultats obtenus avec la méthode développée indiquent qu'elle pourrait bien fonctionner à plus grande échelle, cependant plusieurs des problèmes énoncés dans la section de la revue de la littérature sur l'estimation de la profondeur pourraient compliquer beaucoup l'algorithme en poussant le développement plus loin, notamment sur des images réelles. Par exemple, étant donné que la méthode se base sur un algorithme de recherche de correspondance à base de noyau de recherche, cela implique certains problèmes intrinsèques. Si le déplacement des pixels est plus grand que la fenêtre de recherche, le pixel correspondant trouvé où propager l'in-

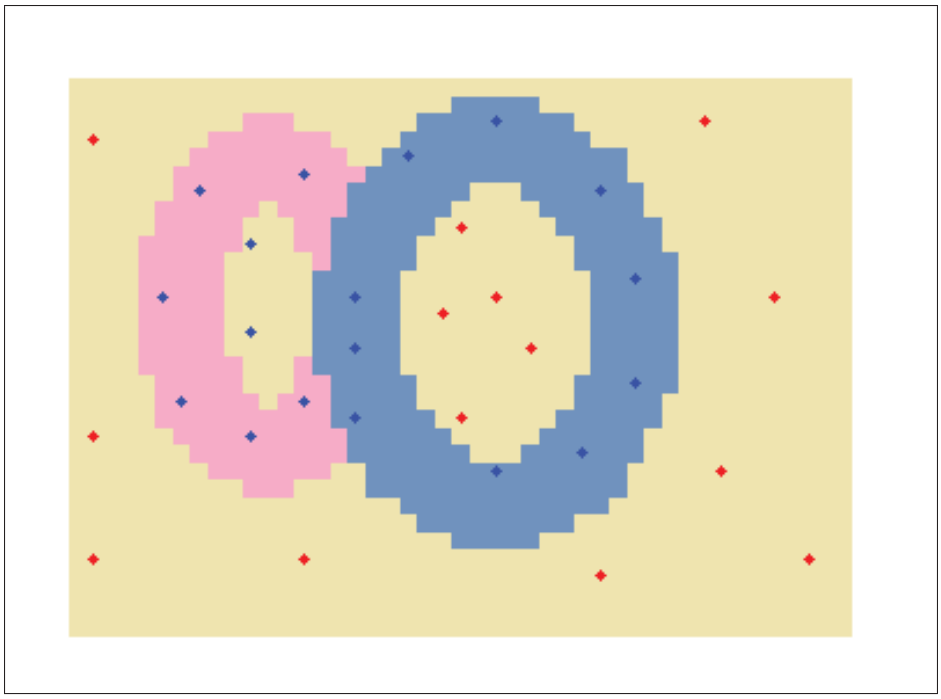

Figure 4.4 Ellipses couleur - annotations

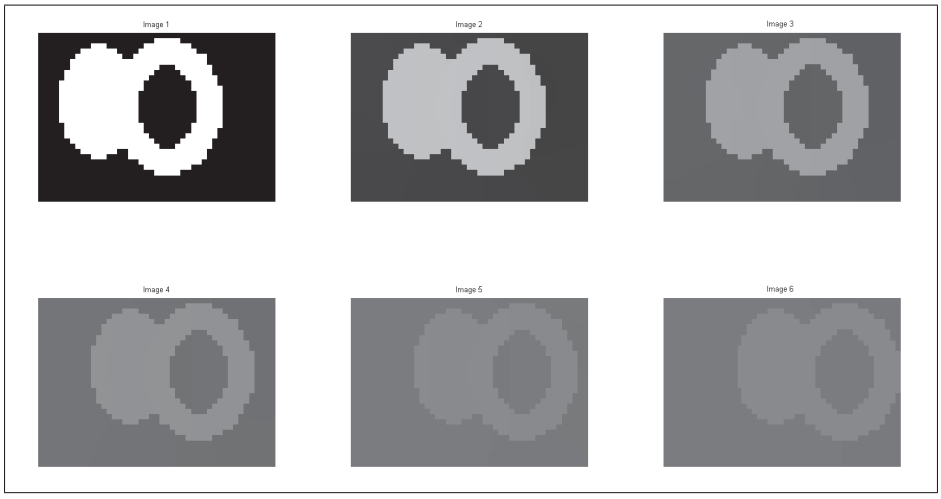

Figure 4.5 Ellipses couleur - Itération 1

formation des annotations ne sera jamais le bon et il en va de même dans le cas des pixels en occlusion. La grande majorité des problèmes potentiels rencontrés par la méthode se retrouvent au niveau de la recherche de la correspondance des pixels entre deux images. Ce besoin d'ob-

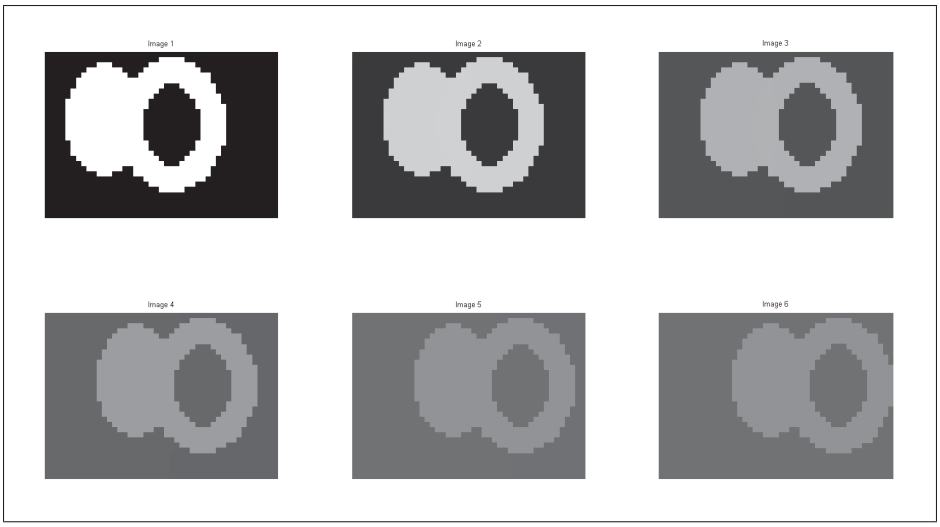

Figure 4.6 Ellipses couleur - Itération 2

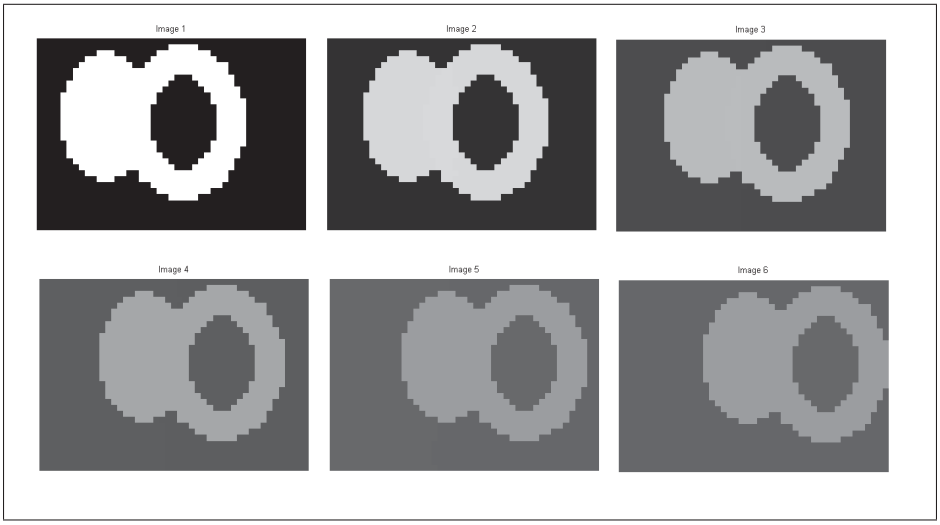

Figure 4.7 Ellipses couleur - Itération 3

tenir une parfaite correspondance est potentiellement réglé par l'estimation de mouvement et c'est ce qui explique l'orientation des recherches subséquentes.

4.2 Améliorations possibles

Plusieurs améliorations ont été pensées pour la méthode mais n'ont pas été explorées davantage puisque le besoin d'une méthode d'estimation de mouvement donnant une correspondance parfaite était trop important.

4.2.1 Méthode d'annotation

Une meilleure méthode d'annotation pour assigner la profondeur a été envisagée, qui permet à l'utilisateur d'annoter avec des traits de pinceau au lieu d'assigner pixel par pixel. Cette amélioration n'a pas été testée pour vérifier si les résultats obtenus étaient plus précis puisque les tests effectués montrent déjà l'efficacité de la méthode avec un très petit nombre de pixels. Cette différente manière d'annoter les images présente une amélioration pour l'utilisateur mais ne modifie pas l'algorithme puisque les pixels annotés sont déjà tous considérés. Une autre amélioration possible sur le même thème était de permettre à l'utilisateur d'effectuer des annotations avec des valeurs non constantes de profondeur. Comme c'est généralement le cas dans des images réelles, les objets ne constituent pas un plan unique de profondeur mais présentent différentes épaisseurs et différentes formations géométriques qui font que la profondeur d'un objet peut changer d'un pixel à l'autre. Cette modification représenterait des changements significatifs dans la méthode pour pouvoir considérer un nombre important de classes de profondeur. Par exemple, il serait possible de considérer une gamme de valeurs de profondeur pour un objet plutôt qu'une seule valeur, mais il serait nécessaire de trouver une façon de spécifier la direction des variations de profondeur ainsi qu'une forme géométrique. Par exemple dans le cas d'une sphère, il faudrait pouvoir spécifier que le point central de la sphère est le plus près de la caméra et que la profondeur augmente plus on se rapproche des rebords. Ce problème pourrait aussi être partiellement résolu en intégrant un système d'interpolation entre les valeurs de profondeur assignées.

4.2.2 Classes de profondeur multiples

Les tests effectués ont tous été réalisés avec seulement deux classes de profondeur par souci de simplicité mais il est évident que pour appliquer la méthode sur des images réelles il est nécessaire d'offrir la possibilité d'annoter avec multiples valeurs de profondeur. Pour considérer un petit nombre de classes de profondeur, cela ne requiert pas de changements importants dans la méthode. Il faut modifier l'algorithme pour que chaque classe soit comparée une fois contre chacune des autres classes et une probabilité d'appartenir à chacune des classes sera calculée pour chaque pixel. Cela réduira probablement la certitude absolue d'assignation de profondeur de chaque pixel puisque l'on considère plus de possibilités qu'avant mais en utilisant des seuils il est possible d'éliminer les probabilités les plus faibles.

Pour travailler avec un nombre élevé de classes de profondeur (par exemple les valeurs maximales de 256 teintes de gris), il faudrait probablement modifier substantiellement la méthode. Il faudrait tout d'abord étudier la perception humaine pour découvrir si vraiment l'humain est capable de discerner 256 niveaux de profondeur dans une image en 3 dimensions. Il serait envisageable de regrouper les différentes valeurs de profondeur dans un nombre n de regroupement de valeurs pour diminuer les possibilités selon le maximum de précision qui peut être discerné par l'œil humain. L'ajout de ces considérations augmenterait de toute façon notablement la complexité des calculs à effectuer.

Cela permettrait de se rapprocher de valeurs de profondeur continues, donc le problème deviendrait une régression plutôt qu'un problème de classification. En transformant aussi la méthode pour obtenir une solution probabiliste plutôt que déterministe, il serait possible d'interpoler des valeurs de profondeur selon une proportion reliée à chaque probabilité. Par exemple si l'on se retrouvait dans un cas où on avait 50% de chance d'avoir une profondeur égale à 1 et 50% de chance d'avoir une profondeur de 2, on se retrouverait avec une profondeur de 1.5.

4.3 Problèmes envisagés

Une méthode basée sur la propagation de l'information comme celle-ci se verra toujours confrontée à des problèmes communs qui devront être résolus d'une manière quelconque. Ces problèmes ont été prévus et des solutions ont été envisagées mais n'ont pas été testées. Certaines des solutions ont été testées dans le cadre de la méthode d'estimation de mouvement puisque la plupart de ces problèmes sont liés à la correspondance entre les pixels que l'estimation de mouvement se veut améliorer. Cette section présente une liste concise de ces problèmes ainsi que des solutions proposées.

4.3.1 Problème des occlusions

Dans une situation où l'on a une séquence vidéo impliquant du mouvement, il va invariablement avoir des occlusions d'une image à l'autre quand des portions d'un objet vont, par exemple, passer derrière un autre objet. Cela veut dire que des pixels d'une image n'auront pas de correspondance dans l'image suivante ou vice-versa. Cela pose problème pour une méthode qui essaie de trouver une correspondance 1 pour 1 entre les pixels des images dans le temps. Cela va généralement créer des problèmes de mauvaise correspondance puisqu'un pixel ne trouvant pas sa correspondance va probablement choisir un des pixels voisins ce qui va causer des problèmes au niveau du déplacement calculé des pixels.

Il y a potentiellement plusieurs solutions à ce problème. Selon la littérature consultée, il n'y a pas de solution miracle adoptée par tous et c'est la raison pourquoi plusieurs solutions ont été envisagées et certaines testées. L'ébauche de solution retenue pour ce problème est d'avoir une classe spéciale étant équivalente au mouvement ou à la profondeur correspondante à une occlusion. Cette solution implique une technique pour trouver où sont les occlusions ce qui est parfois plus simple que de régler le problème des occlusions. Dans cette solution, les occlusions sont trouvées en exécutant la correspondance entre les deux images en allant de l'avant et en allant par l'arrière. En effectuant cette comparaison dans les deux sens, par la suite les mouvements calculés de chaque pixel sont comparés. Les pixels ayant des vecteurs de mouvement

opposés dans les deux sens sont validés comme étant de bons résultats, par contre les pixels ayant de grandes différences sont de bons candidats pour des occlusions, encore plus dans le cas où ces pixels sont placés à la bordure d'un objet. Une fois ces pixels ciblés il est possible de les marquer comme des occlusions puis d'appliquer une méthode différente pour assigner la profondeur.

4.3.2 Problème de l'estimation de mouvement

Comment mentionné plus haut, cette méthode dépend fortement d'établir une bonne correspondance entre chaque pixel d'une image et les pixels de l'image suivante. Étant donné qu'une séquence d'images réelles est rarement statique, les objets et donc les pixels vont se déplacer dans le temps, ce qui implique que pour la propagation de l'information, par exemple entre l'image clé annotée et l'image suivante, il est impératif de trouver avec précision le déplacement des pixels d'une image. Si l'on a associé une profondeur à un objet, on doit pouvoir associer la même profondeur à un objet dans l'image suivante. La section de l'état de l'art note plusieurs techniques pour produire une bonne estimation de mouvement, cependant après réflexion et expérimentations, il s'est avéré que la technique du *random-walker* employée pour la propagation de l'information pouvait tout aussi bien être appliquée au problème d'estimation de mouvement puisque son principe de base s'y prête bien. C'est ce besoin d'une bonne technique d'estimation de mouvement ainsi que la perspective des problèmes que l'on allait rencontrer avec la méthode qui a réorienté les recherches vers l'estimation de mouvement. Toujours en voulant améliorer la technique d'estimation de profondeur avec une méthode plus poussée d'estimation de mouvement, c'est dans cette direction que les recherches ont continué.

4.3.3 Mouvement hors du plan de la caméra

Un problème envisagé aussi mais pour lequel aucune solution n'a été conçue surviendrait quand les objets ont un mouvement de grande envergure dans l'axe de la caméra. C'est-à-dire qu'ils se rapprochent ou s'éloignent de la caméra. Essentiellement, le problème vient du fait que l'objet va contenir plus ou moins de pixels en prenant plus ou moins de place dans l'image.

Ceci va à l'encontre de la contrainte de consistance du mouvement intra-image puisque les différents pixels d'un même objet auraient des mouvements différents et que cela va à l'encontre de cette contrainte. Ce problème a été envisagé comme posant une difficulté potentielle mais des tests de ce cas n'ont pas été étudiés en profondeur pour permettre de suggérer une solution potentielle autre que de relaxer la contrainte mentionnée précédemment.

4.3.4 Régions non texturées

Un autre problème fréquemment rencontré par les méthodes calculant une correspondance de pixel à un niveau très précis est le manque d'information pour différencier les pixels de ses voisins. Étant donné que pour trouver le pixel correspondant au pixel courant dans l'autre image on le compare à plusieurs pixels dans une région rapprochée à sa position originale, si chacun de ces pixels est identique, il devient assez difficile de trouver la correspondance exacte.

CHAPITRE 5

RÉSULTATS OBTENUS - ESTIMATION DE MOUVEMENT

Des tests plus exhaustifs ont été effectués pour la méthode d'estimation de mouvement puisque la recherche s'est concentrée sur le sujet. Il existe aussi des méthodes et des ensembles d'images distribuées pour quantifier le succès d'une méthode d'estimation de mouvement. Par exemple, il existe des bancs de tests qui permettent de classer la méthode par rapport à plusieurs méthodes publiées. Il est possible de classer une méthode selon son efficacité en comparant les résultats obtenus avec des résultats de référence, il en existe plusieurs mais ce qui a été utilisé majoritairement est *Middleburry* [ref : Middleburry]. Les premiers tests ont été effectués visuellement en affichant des flèches sur chaque pixel pour visualiser le mouvement calculé pour chaque pixel. Une autre technique qui a été utilisée est d'attribuer une gradation de couleur pour chaque mouvement possible et de superposer une image de couleur sur l'image étudiée pour visualiser le mouvement. Chaque sous-section suivante présentera les résultats obtenus, séparés selon la méthode utilisée pour visualiser qui correspond aussi à un certain ordre chronologique d'évolution de la méthode. Même si les résultats sont comparés visuellement, les explications utiliseront les deux mesures qui sont utilisées par Middleburry pour comparer les méthodes, elles sont définies dans les sections suivantes.

5.1 Représentation par flèches / Comparaison visuelle

Les premiers tests effectués pour évaluer le potentiel de la méthode n'étaient pas encore comparés à des résultats de référence mais étaient jugés visuellement, souvent parce que le déplacement dans l'image était relativement simple à représenter. En premier lieu, la direction du mouvement trouvé par la méthode était considérée comme le facteur de réussite le plus important. Puisque la précision de la méthode est limitée au pixel près, il est moins grave d'avoir une erreur entre un déplacement trouvé de (1, 0) et un déplacement réel de (2, 0) que de trouver un déplacement de (-1, 0) pour le même déplacement réel. Premièrement, mathématiquement cela représente une plus petite différence de distance absolue et, finalement, pour une application

qui utiliserait ensuite le mouvement trouvé, un mouvement dans la bonne direction mais de plus petite envergure génère moins d'erreurs potentielles qu'un mouvement dans la mauvaise direction. Les premiers tests effectués ont surtout servi à valider la faisabilité de la méthode et à donner une évaluation sommaire de la performance de la méthode.

Des tests volontairement plus faciles à réaliser ont été effectués. En général, les images étaient fortement texturées autant au niveau du fond de l'image que de l'objet qui se déplaçait dans la séquence. Les objets étant souvent des ellipses aussi fortement texturées ce qui permettait à la méthode de facilement la distinguer par rapport au fond de l'image qui était immobile. Certaines formes différentes et moins régulières ont été utilisées comme vérificateur de la méthode tout au long des expérimentations pour vérifier que la méthode pouvait toujours résoudre des cas simples, même après des changements significatifs de la logique employée. Il est difficile de montrer des images complètes puisque les résultats présentent une flèche par pixel et les images employées sont de relativement grande taille pour permettre de tester aussi la performance de la méthode en terme de temps d'exécution. Ce qui est plutôt important est de visualiser certaines parties de l'image où il peut y avoir des erreurs, notamment dans les cas mentionnés précédemment où il peut y avoir des occlusions, des textures ne fournissant pas assez d'information pour la différenciation et dans le cas de mouvements non linéaires.

Les résultats obtenus sur ces images de tests montrent que la méthode proposée peut au moins résoudre des problèmes simples et c'était le but du test effectué. Ces tests permettent de valider tout changement à la méthode comme étant un changement positif et permettent aussi souvent de déceler des problèmes qui pourraient survenir lors d'une modification. Par exemple si dans les résultats obtenus sur ces tests simples on commence à remarquer des frontières plus floues, on peut en déduire que, probablement, la dernière modification ne respecte pas très bien les contraintes de similarité entre les pixels voisins. À plusieurs reprises durant les expérimentations, ces images ont permis de corriger un problème car elles rendent plus facile l'analyse des erreurs.

5.2 Représentation par couleurs

Une autre technique de visualisation et de comparaison des résultats obtenus par la méthode avec des résultats de référence est de créer une distribution de couleurs représentant chaque mouvement possible dans un plan (x, y). Cette façon de procéder peut rendre la comparaison à première vue plus facile puisqu'elle permet une vue d'ensemble rapide sur les résultats obtenus. Le processus de génération d'un tel ensemble de couleurs implique généralement de transformer les coordonnées des vecteurs de mouvements (x, y) en une valeur de couleurs RGB. Une façon simpliste est de prendre la valeur minimale en x (soit -taille horizontale de la fenêtre de recherche) et de la faire correspondre à une petite valeur entre 0 et 255 en rouge. Ensuite, on divise les valeurs jusqu'à 255 en sauts égaux correspondants au nombre possible de variations de distance horizontale. On fait ensuite de même pour les déplacements verticaux avec la couleur bleue et on combine ensuite les deux pour obtenir une image de référence pour visualiser les mouvements.

Plusieurs tests ont été réalisés en utilisant cette technique de visualisation puisque l'ensemble d'images qui ont été utilisées pour mesurer l'efficacité de la méthode fournit un espace de couleurs semblablement construit. Pour visualiser rapidement l'amélioration ou la dégradation de la méthode suite à des changements il suffit de comparer visuellement l'image de référence de mouvements avec celle générée par la méthode. Cette technique permet aussi facilement de discerner les zones d'erreur importantes dans les résultats obtenus en se fiant à la teinte de la couleur attribuée à la zone de pixels, une teinte très différente signifie probablement une erreur importante de direction des vecteurs de mouvement calculés ce qui peut nécessiter une investigation pour comprendre la raison de l'erreur produite. Cette façon de procéder a permis souvent de régler des problèmes présents dans les modèles utilisés par la méthode et permet aussi de rapidement ajuster les paramètres qui contrôlent les résultats obtenus en fonction de l'image colorée des vecteurs de mouvements obtenus. Voici un aperçu de l'espace de couleurs utilisé pour visualiser les résultats [Middleburry].

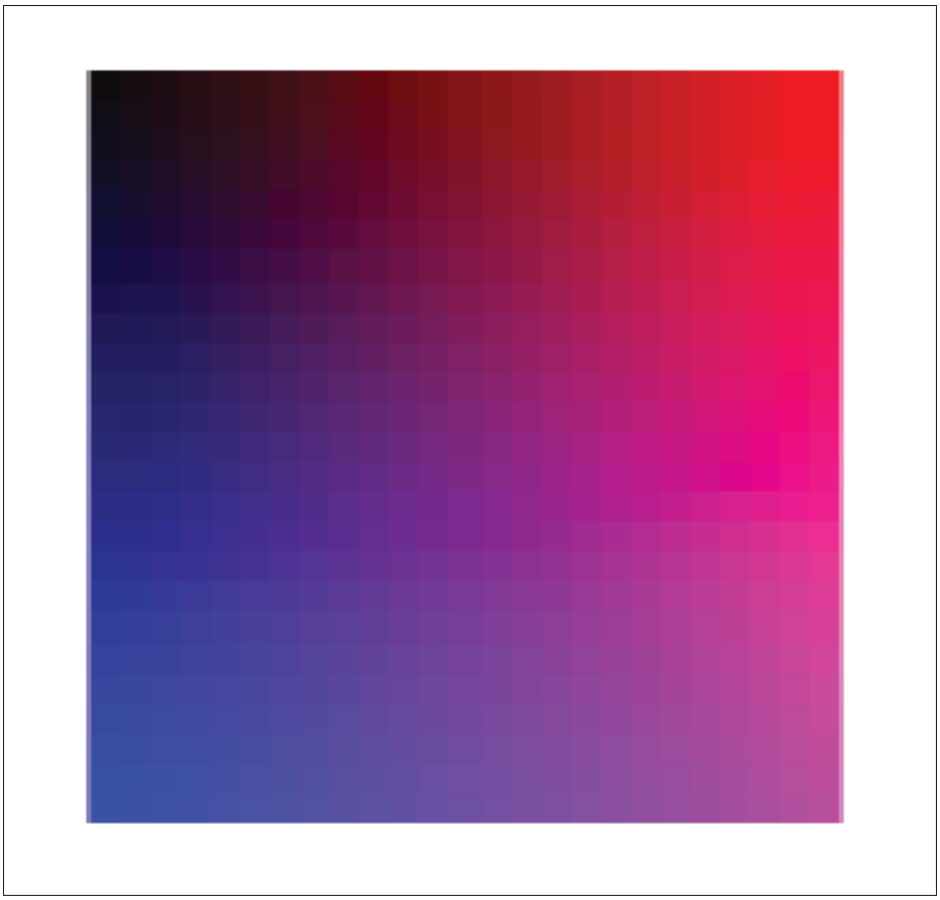

Figure 5.1 Carte de couleurs pour taille 25

La technique pour afficher un flux optique en fonction de l'espace de couleur est aussi fournie par Middleburry, donc en générant deux matrices représentant les déplacements horizontaux et verticaux on arrive à générer une image colorée en fonction des déplacements. Cependant, l'espace de couleur utilisé par Middleburry est continu mais les déplacements générés par la méthode sont distincts alors il se glisse parfois des erreurs d'arrondis qui peuvent faire apparaître la zone comme étant en erreur alors qu'en fait c'est seulement une question d'arrondi qui a généré une teinte différente. C'est pour cette raison qu'il est important de toujours garder la carte complète des couleurs pour faire la comparaison et non pas seulement comparer la teinte visuellement.

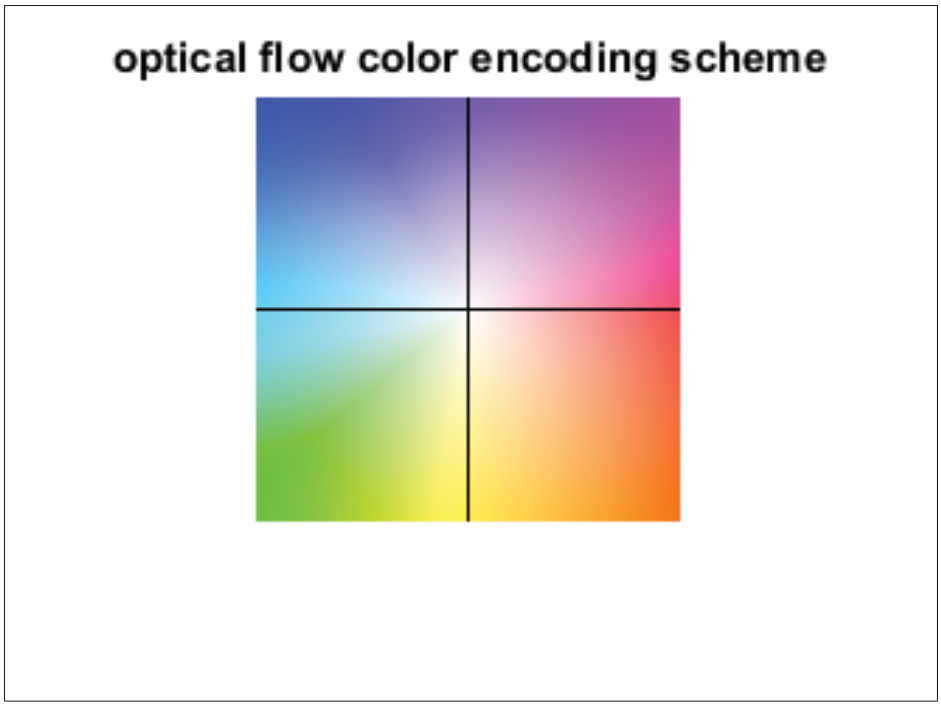

Figure 5.2 Espace de couleurs du mouvement continu fourni par Middleburry

5.3 Différence de position finale *endpoint error*

La première métrique employée pour comparer les techniques est l'erreur de position finale ou *endpoint error*. Elle représente la différence moyenne géométrique entre la position finale calculée pour chaque pixel et la position réelle définie par l'image de référence contenant le mouvement exact de chaque pixel. En d'autres mots, elle est définie comme suit :

$$AEE = \frac{1}{NM} \sum_{ij}^{NM} \sqrt{(x_{ij} - GT_{ij})^2} \tag{5.1}$$

Où AEE = *Average Endpoint Error*, pour une image de taille N par M, où x_{ij} correspond à la position du pixel à la rangée i et colonne j et GT_{ij} dénote la valeur de référence de déplacement de pixel fournie par le contenu d'évaluation. Cette métrique sert principalement à mesurer l'ampleur moyenne de l'erreur en distance de pixels, c'est-à-dire que plus le mouvement calculé amène le pixel loin de la position de référence, plus l'erreur sera importante.

La magnitude de cette métrique est plus importante selon la magnitude des mouvements dans l'image et c'est en présence de grands déplacements qu'elle permet le mieux d'évaluer la précision d'une méthode. Dans des cas ou le mouvement est de petite envergure, l'erreur calculée peut être trompeuse, par exemple dans le cas où le mouvement calculé serait de (-1, 0) et que le mouvement de référence est de (1, 0), l'erreur est seulement de 2 pixels, mais en pratique le mouvement estimé part complètement dans la direction opposée. C'est pour cette raison qu'une métrique différente est aussi calculée en parallèle.

5.4 Différence d'angle de mouvement *Angular Error*

Complémentaire à la métrique de différence de position, la différence d'angle de mouvement calcule la moyenne des différences d'angles entre le mouvement calculé et l'angle véritable du mouvement. Elle est définie comme le produit scalaire des deux vecteurs divisé par le produit de leurs normes et ensuite on calcule le cosinus inverse (**arccos**). Les deux vecteurs sont nommés **MV** pour le vecteur calculé et **GT** pour la réponse attendue. La métrique est appliquée sur chaque pixel et la moyenne sur toute l'image est calculée pour avoir une mesure d'erreur d'angle moyenne. Pour tenir compte du fait que les vecteurs peuvent être nuls, les vecteurs sont tous considérés sur trois dimensions et ayant une valeur unitaire pour la coordonnée en Z.

$$AAE = \arccos\left(\frac{MV \cdot GT}{(\|MV\| \cdot \|GT\|)}\right) \tag{5.2}$$

Par opposition à l'erreur moyenne de position, l'erreur angulaire est plus efficace sur les petits déplacements puisque c'est l'angle entre les deux vecteurs qui compte plutôt que la distance entre leur point final. La faiblesse de cette métrique est qu'elle n'arrivera pas à calculer une différence importante entre deux vecteurs de mouvement ayant le même angle mais une distance différente. La conclusion de l'étude de ces deux erreurs est qu'elles ne peuvent pas tout à fait être remplacées et devraient être utilisées en conjonction l'une avec l'autre. Pour appuyer cette conclusion, la table de comparaison des résultats de *Middleburry* offre la classification

des méthodes en fonction de ces deux métriques même s'ils semblent proposer d'utiliser prioritairement l'erreur de différence de position finale (EE).

5.5 Images artificielles

Parmi les images fournies par *Middlebury*, il y a une série d'images qui correspond bien aux tests à effectuer avec la méthode. Ces images fournissent les vecteurs de mouvements en référence pour permettre d'évaluer sa méthode et plusieurs autres méthodes sont aussi classées sur le site du *data set* selon les résultats obtenus en affichant les métriques utilisées comme l'erreur angulaire moyenne et l'erreur de mouvement moyenne. C'est avec cette séquence d'images que la majorité des tests de performance de la méthode ont été réalisés. Dans ces images, on retrouve les images **Urban 2** et **Urban 3** qui représentent des scènes créées artificiellement contenant des gratte-ciels et qui présentent généralement un mouvement de caméra constant.

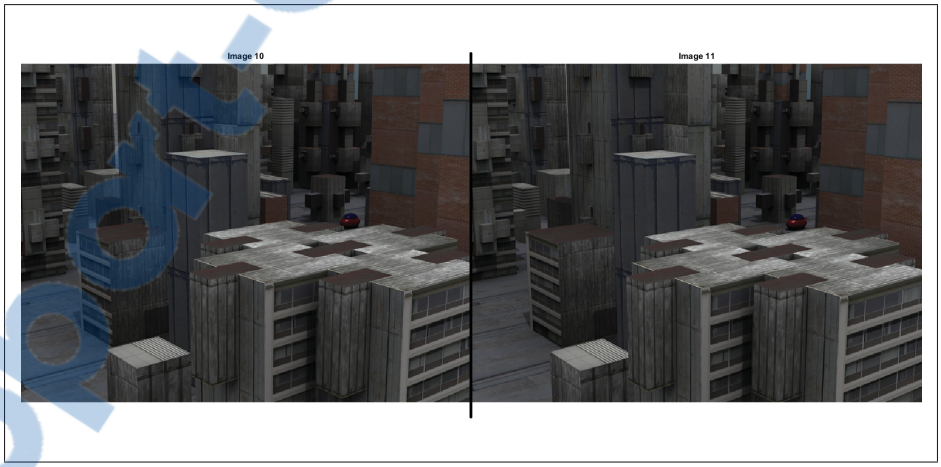

Figure 5.3 Images 10 et 11 - Middleburry Urban 2

Ces images ont été choisies puisque les vecteurs de mouvements de référence sont fournis par *Middleburry* pour pouvoir tester les méthodes contre la réponse à obtenir. Ces images sont 640x480 pixels et contiennent 3 canaux de couleurs **RVB** (Rouge-vert-bleu). Elles sont aussi intéressantes d'un point de vue de l'estimation de mouvement car la scène est composée d'objets solides qui ne se déforment pas avec le mouvement et qui sont fortement texturés. Ces

caractéristiques répondent bien aux demandes de la méthode, particulièrement, des objets se déformant ne sont pas bien supportés par la méthode. Selon la conception de la méthode, des pixels similaires doivent normalement se déplacer dans la même direction puisque les pixels voisins semblables vont s'influencer entre eux pour aller dans la même direction. Aussi, comme mentionnées plus haut, de grandes zones sans texture dans une image sont problématiques pour l'estimation de mouvement car on perd de l'information de référence si tous les pixels sont similaires dans une grande zone. Dans ces images, le mouvement vient entièrement de la caméra ce qui implique que les objets bougent généralement dans des directions constantes.

La figure 5.4 représente les résultats obtenus par la méthode pour ces images avec les meilleurs paramètres possible. Pour obtenir un meilleur résultat, on réduit la taille de l'image originale pour réduire l'importance de certaines des particularités de cette image, par exemple il y a plusieurs zones dans cette image très fortement texturées, mais où la texture est très répétitive. Ceci pose un problème pour la méthode puisque pour trouver un pixel similaire dans l'image suivante, on compare les caractéristiques d'un pixel avec tous les pixels dans la fenêtre de recherche dans l'image suivante. Dans ses caractéristiques, on retrouve les voisins immédiats d'un pixel puisque chercher un pixel de la même couleur n'est pas nécessairement suffisant pour trouver le même pixel dans l'image suivante. Dans le cas d'une image ayant beaucoup de textures répétitives, la certitude d'un pixel est diminuée pour chaque pixel ayant des voisins suivant le même motif. Cela a tendance à produire des résultats moins bien distribués puisque la propagation de l'information est influencée par la certitude des pixels voisins.

On remarque qu'en se comparant avec l'image de référence (Figure 5.4a), il y a une petite zone en erreur (en bleu). Cette couleur est attribuée aux déplacements vers le haut à gauche alors que dans l'image de référence (turquoise), le déplacement est plutôt vers la gauche et quelque peu vers le bas. Dans les images originales, cette zone est une forte zone d'occlusions, avec le mouvement de la caméra, la construction en avant-plan passe devant la construction en arrière-plan ce qui cause les erreurs que l'on voit dans l'image. Si on regarde l'image représentant la confiance liée au déplacement, on s'aperçoit que cette zone n'est pas du tout en confiance. Cela implique en fait que plusieurs déplacements possibles étaient également probables dans

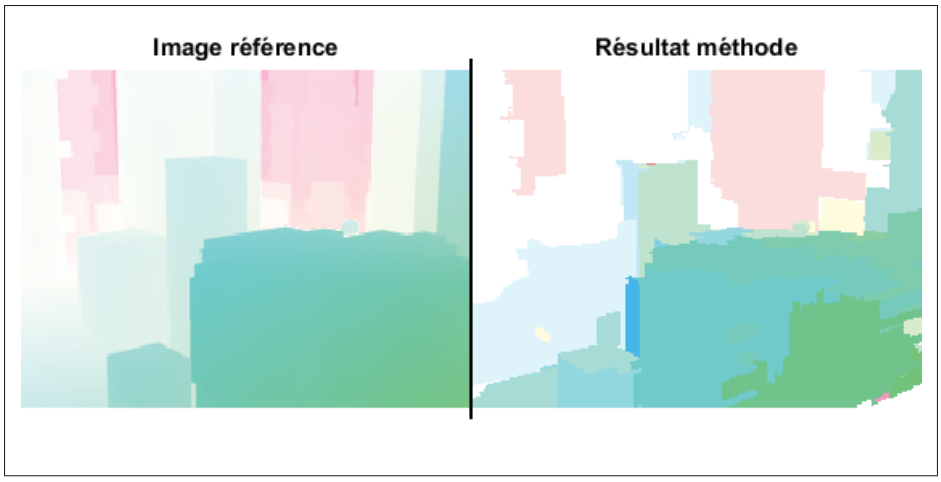

Figure 5.4 Résultats Urban2 - 35% taille réelle

cette zone alors l'algorithme a choisi un déplacement parmi les plus probables. Dans ces cas, il arrive parfois que l'algorithme choisisse le mauvais déplacement ce qui cause des erreurs. Mais comme la confiance est prise en compte dans l'influence qu'ont les pixels sur leurs voisins, ces pixels ayant une basse confiance n'auront pas beaucoup d'influence sur leurs voisins. C'est pourquoi cette zone d'erreur est limitée comme on peut le voir. Dans cette image de confiance (figure 5.5), la teinte passe de noir (moins certain) à blanc (plus certain).

L'image suivante est nommée **"Urban 3"** et fait partie de la même série d'images fournies par *Middlebury*. Elle représente encore une scène dans une ville créée artificiellement et contient encore un mouvement de caméra uniforme. C'est avec cette image que la majorité des tests ont été réalisés.

En analysant l'image colorée obtenue (Figure 5.7) on remarque certaines erreurs assez visibles, par exemple la zone complètement rouge dans le haut de l'image ce qui correspondrait à un mouvement totalement vers la droite. À première vue, cela peut sembler une erreur importante mais le mouvement dans cette zone est bel et bien vers la droite mais moins important que ce que la méthode a trouvé comme étant le mouvement le plus probable. En fait, les résultats obtenus ressemblent beaucoup aux résultats de l'image de référence si ce n'est que des petites

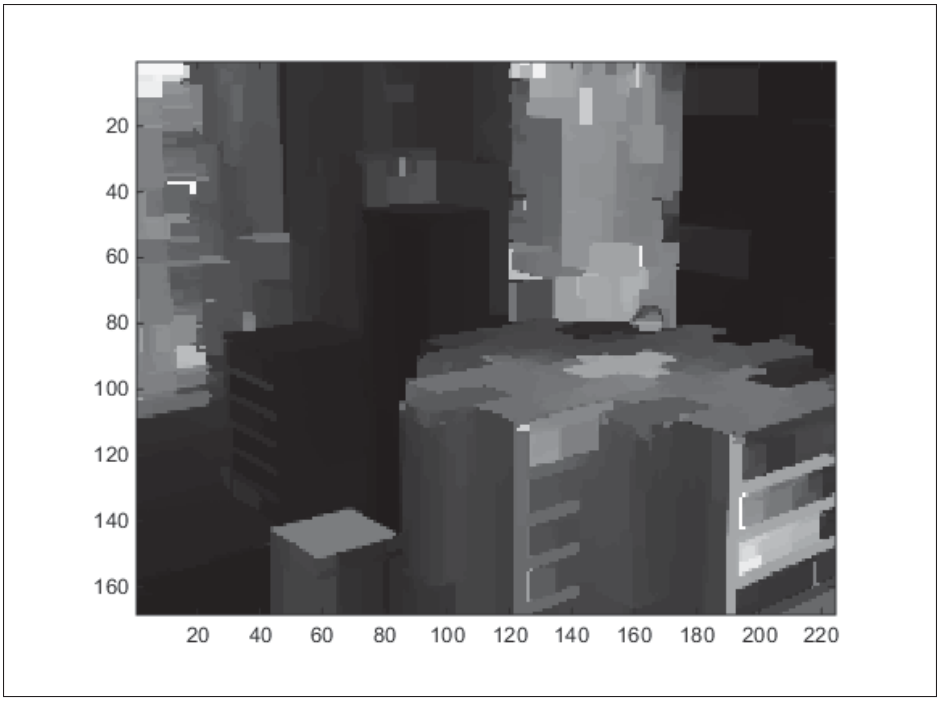

Figure 5.5 Résultats Urban2 - Confiance

Figure 5.6 Images 10 et 11 - Middleburry Urban 3

erreurs de distance ou d'angle qui sont généralement attribuables à une des difficultés listées plus haut.

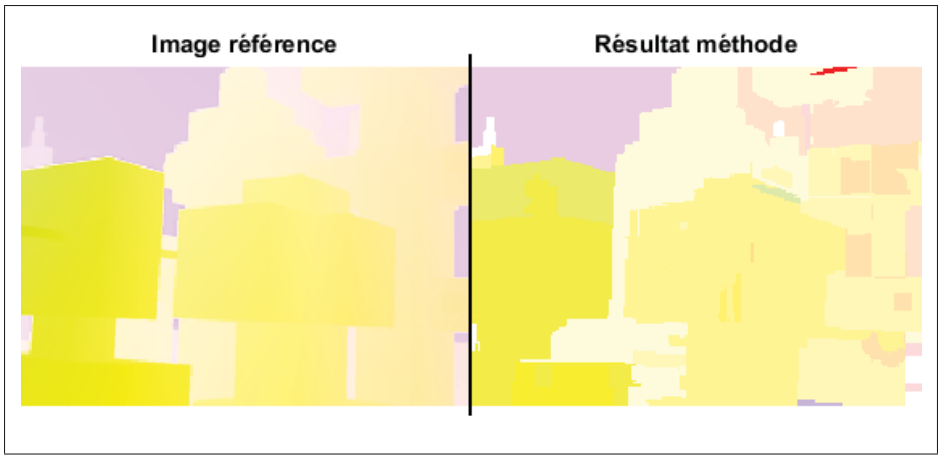

Figure 5.7 Résultats Urban3 - 35% taille réelle

5.6 Effets de la variation des différents paramètres

5.6.1 Paramètre contrôlant la diffusion

Les graphiques suivants représentent les différents résultats obtenus aux métriques d'erreur selon la variation du paramètre beta en conservant les autres paramètres constants. Les valeurs des paramètres d'importance des parties du descripteur ont été obtenues empiriquement en faisant varier les paramètres à travers plusieurs itérations et en sélectionnant les meilleures valeurs qui donnent les erreurs moyennes les plus faibles. Pour les figures suivantes les valeurs suivantes des paramètres Gamma ont été utilisées : [10, 225, 225, 150, 150] ce qui signifie que pour obtenir les meilleurs résultats, la composante L du format de couleurs LAB est beaucoup moins importante que A et B et on accorde une forte importance aux gradients en x et en y.

On remarque qu'en faisant varier le paramètre Beta, on améliore les résultats jusqu'à un certain point et par la suite les résultats deviennent de moins en moins bons. C'est donc qu'il faut pour les différentes images trouver cette valeur du paramètre de diffusion optimale qui va donner les meilleurs résultats. La différence est assez grande entre les valeurs du paramètre beta puisque l'on passe d'une erreur angulaire moyenne de 20 à 6.43 selon les différentes valeurs du paramètre de diffusion.

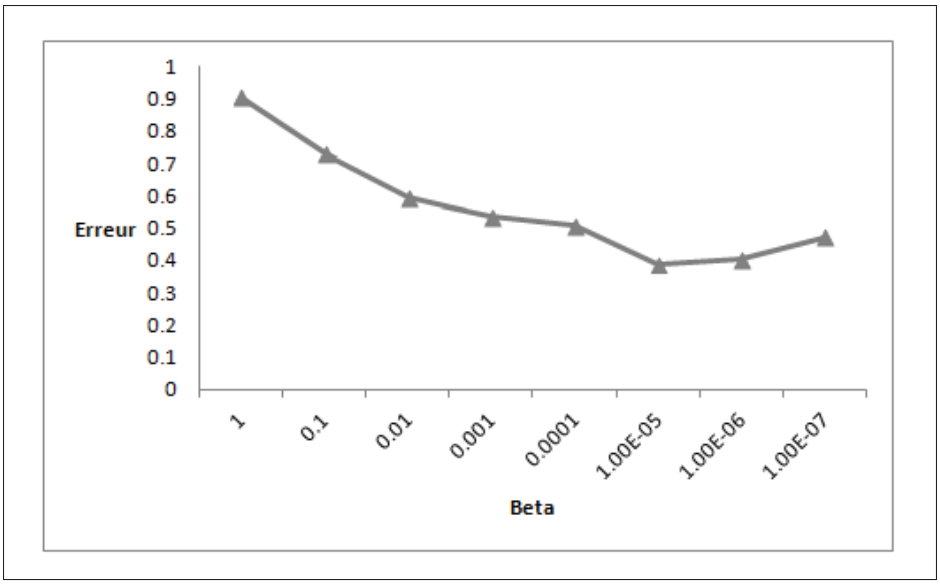

Figure 5.8 Erreur de distance moyenne selon Beta

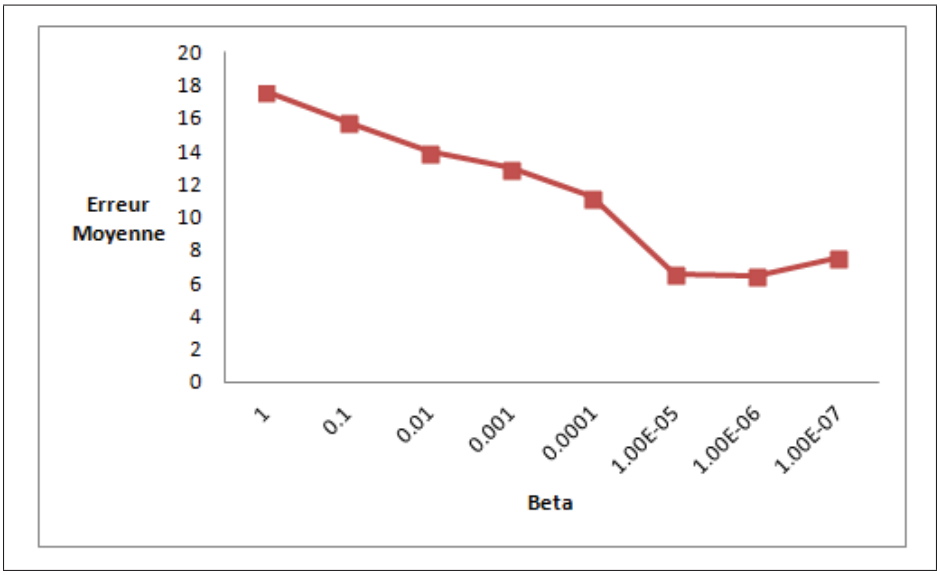

Figure 5.9 Erreur d'angle moyen selon Beta

La figure 5.10 représente les résultats obtenus sur l'image Urban 3 avec le paramètre de diffusion beta étant 1. Ces images symbolisent la correspondance initiale obtenue en n'appliquant aucune diffusion de probabilité de mouvement dans l'image. Pour ces images, les correspondances fournies par SIFT ne sont pas utilisées. Et le calcul de mouvement est effectué sur chaque pixel de l'image (c'est-à-dire que les SP ne sont pas utilisés). La figure 5.11 montre les

résultats obtenus sur les images de Middlebury Urban 2. On remarque déjà une bonne distinction des formes générales contenues dans l'image. Ce qui est important d'obtenir est une bonne distinction générale des zones de mouvements et que les mouvements obtenus soient dans la bonne orientation.

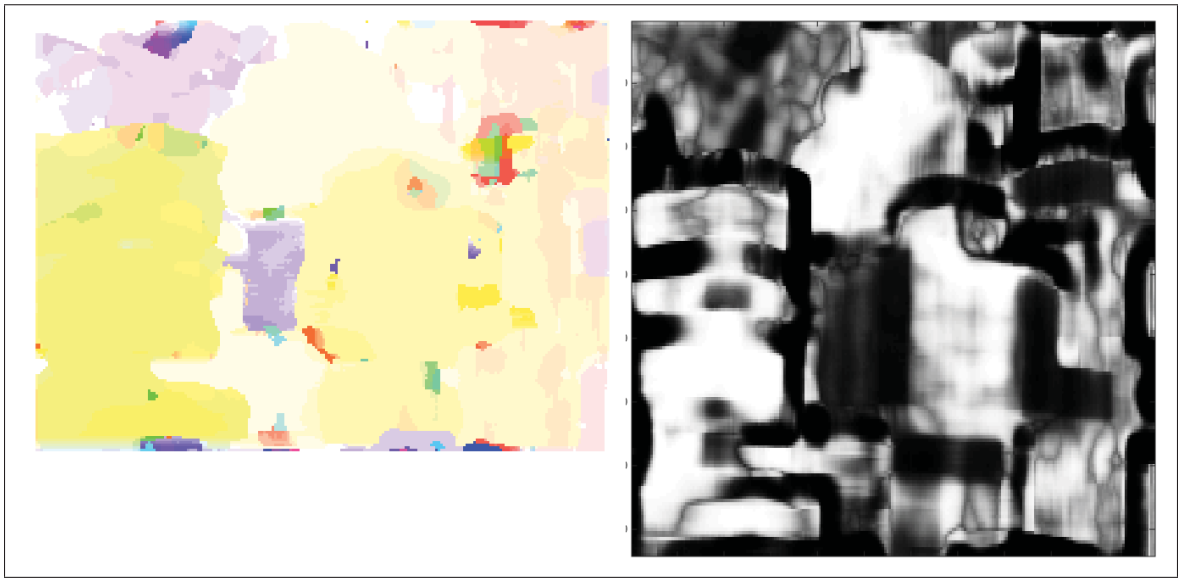

Figure 5.10 Middlebury Urban3
a. Mouvement représenté par couleurs (beta = 1)
b. Certitude du résultat

5.6.2 SIFT

En ajoutant la correspondance SIFT à la méthode, cela permet d'utiliser la force de SIFT qui est de trouver des pixels correspondants ayant une très grande certitude entre les deux images et d'utiliser ceux-ci pour influencer fortement les pixels voisins en utilisant la diffusion de l'information. Principalement, l'ajout de SIFT à la méthode sert à s'assurer que la certitude de certaines des correspondances trouvées est assez élevée pour influencer les autres pixels avoisinants mais aussi pour renforcer les pixels déjà déterminés. Pour déterminer l'impact d'ajouter l'utilisation de SIFT à la méthode, il faut fixer le paramètre β et solutionner le problème sans SIFT puis avec et comparer les résultats obtenus.

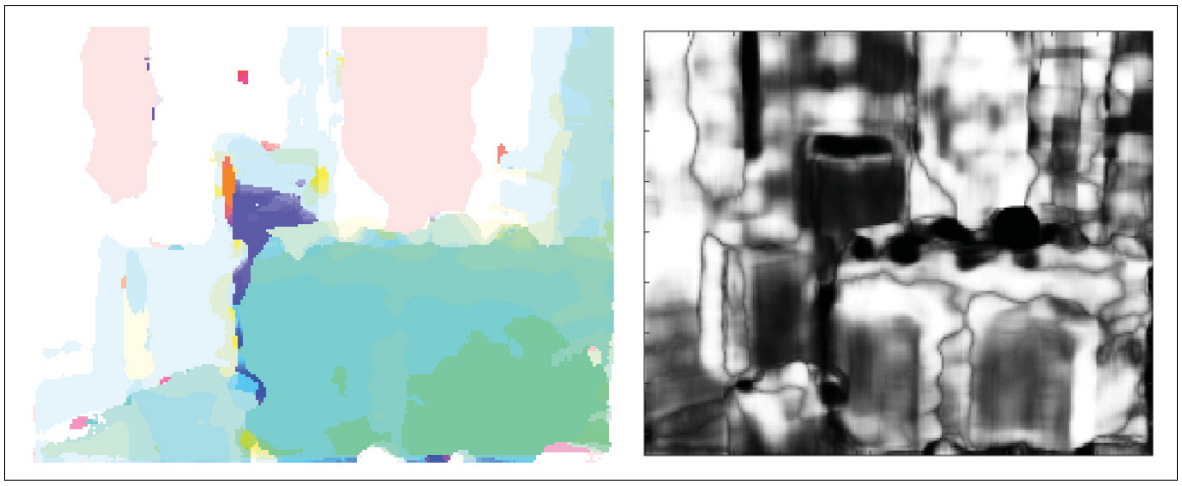

Figure 5.11 Middlebury Urban2
a. Mouvement représenté par couleurs (beta = 1)
b. Certitude du résultat

Sans l'utilisation de SIFT et en ayant fixé la valeur de β à 0.01, la différence de position finale moyenne est de 0.67893 et la différence d'angle de mouvement moyenne est de 12.9142. Avec l'utilisation de SIFT et la même valeur de β, la différence de position finale moyenne est de 0.65867 et la différence d'angle de mouvement moyenne est 12.4708. Le comparatif des résultats est visible à la figure 5.12.

5.6.3 Taille du noyau

La taille du noyau dans la méthode contrôle la fenêtre de recherche qui est utilisée pour explorer l'image suivante pour rechercher une correspondance pour un pixel. La taille du noyau de recherche pour l'image suivante et celle qui détermine dans la même image la taille affectée du voisinage sont deux paramètres séparés ce qui permet un contrôle plus précis sur les résultats obtenus par la méthode. Dans la plupart des expérimentations, la taille du noyau de recherche dans la même image a été fixée à 3 ce qui signifie que 8 pixels autour du pixel recherché sont considérés. Et pour ce qui est de la taille du noyau dans l'image suivante, elle peut varier en fonction de la taille de l'image. Elle doit être assez grande pour englober tout mouvement qui pourrait possiblement survenir entre deux images d'une séquence. Il faut aussi prendre en compte que la taille originale de l'image sera aussi diminuée souvent jusqu'à 35% de sa taille

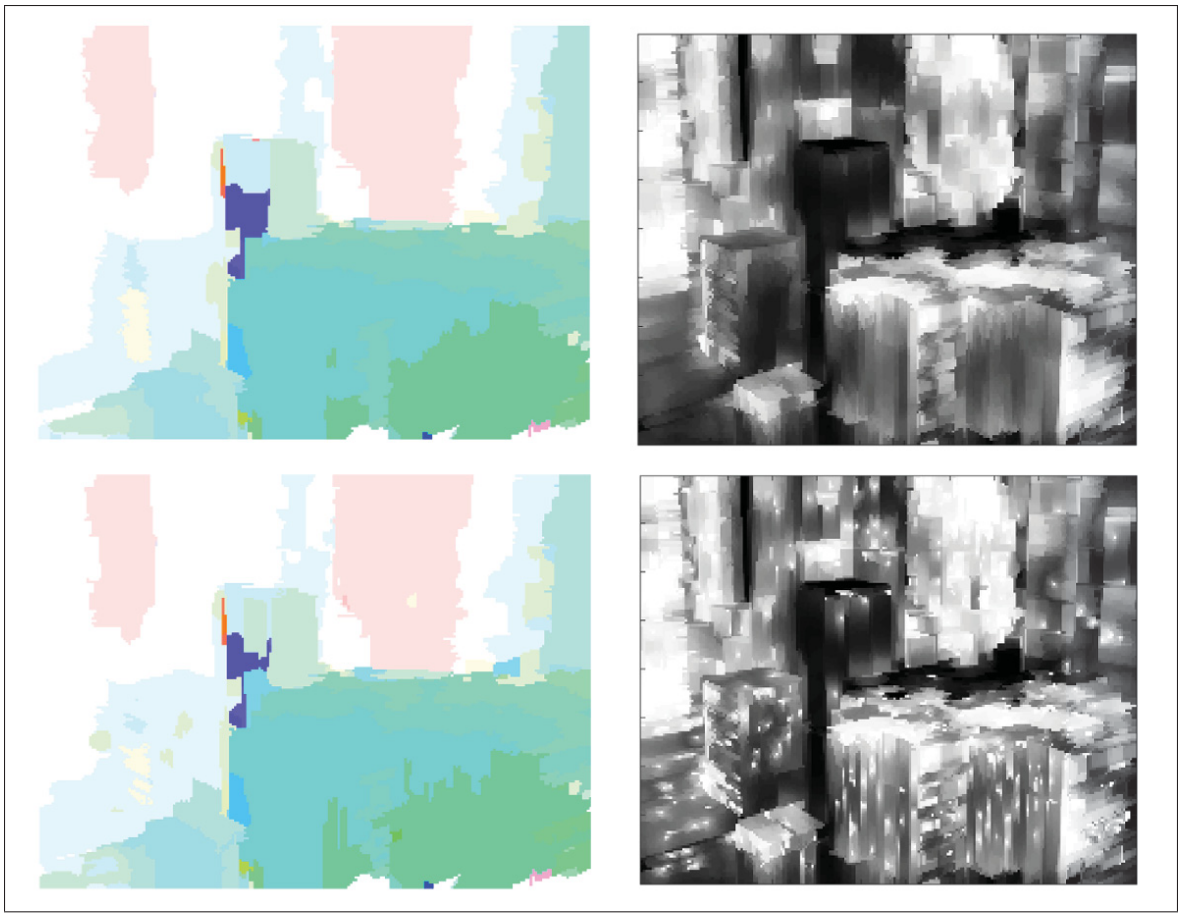

Figure 5.12 Comparaison avec et sans SIFT avec $\beta = 0.01$
Ligne du haut sans SIFT
Ligne du bas avec SIFT

originale. Durant les expérimentations, une taille de noyau entre 10 et 15 semble donner les meilleurs résultats adaptés à la taille de l'image utilisée.

En prenant en exemple l'image *Urban3* et en fixant le paramètre de diffusion β à 0.0001 avec une taille de noyau inter-images à 5, on obtient comme résultat ce qui est présenté à la figure 5.13. Les images 1 et 2 présentent un cas où le noyau de recherche serait trop petit pour capturer l'amplitude de tous les mouvements contenus dans l'image. Les images 3 et 4 présentent les résultats quand tous les mouvements sont capturés. Cependant, la qualité des résultats obtenus vient au coût d'un temps de calcul plus élevé puisque pour générer la matrice de similitude U, il faut considérer 15 par 15 donc 225 pixels au lieu de 5 par 5 (25). Augmenter la taille du noyau

de recherche davantage ne donnerait pas de meilleurs résultats puisque les mouvements sont déjà inclus parmi le noyau. Cela ne ferait qu'augmenter le temps de calcul et aussi diminuer potentiellement la certitude de chaque mouvement puisqu'il existerait une plus grande quantité de mouvements possibles pour chaque pixel.

Figure 5.13 Comparaison sur la variation de la taille du noyau de recherche (Avec $\beta = 0.0001$)
Ligne du haut : Avec taille=5, AEE=1.4848, AAE=23.981
Ligne du bas : Avec taille=15, AEE=0.70039, AAE=15.1349

5.6.4 Superpixels

Les résultats suivants présentent les différentes mesures d'erreurs obtenues en fonction de plusieurs des paramètres de contrôle de la méthode. La première figure 5.14 présente les AEE en

fonction du paramètre β quand les SP sont utilisés pour solutionner le problème et la seconde l'AAE. Comme pour la méthode sans les SP, on remarque qu'à un certain point augmenter le paramètre de diffusion donne des moins bons résultats, il faut donc trouver la bonne valeur de β pour obtenir les meilleurs résultats.

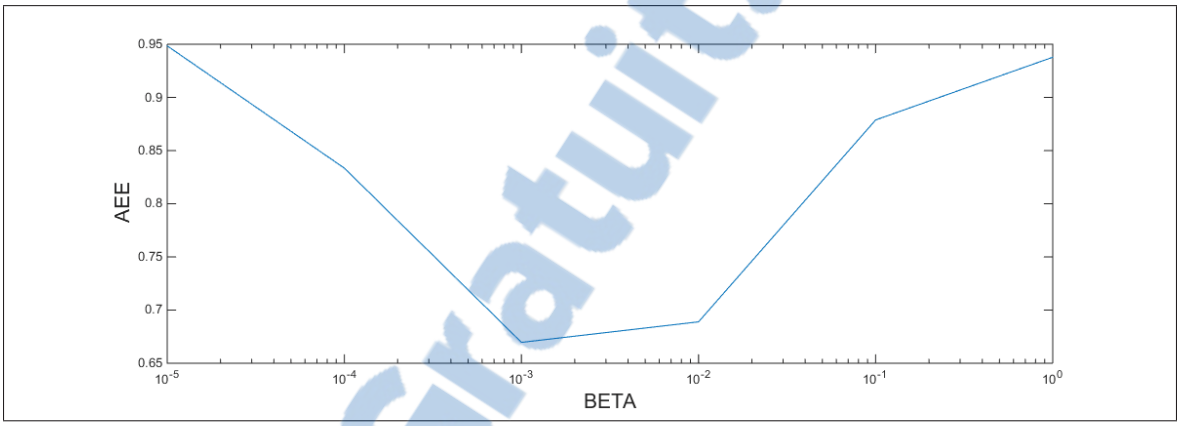

Figure 5.14 Comparaison de l'effet de la variation de β sur les SP (AEE)
maximum : 0.94846
minimum : 0.66964

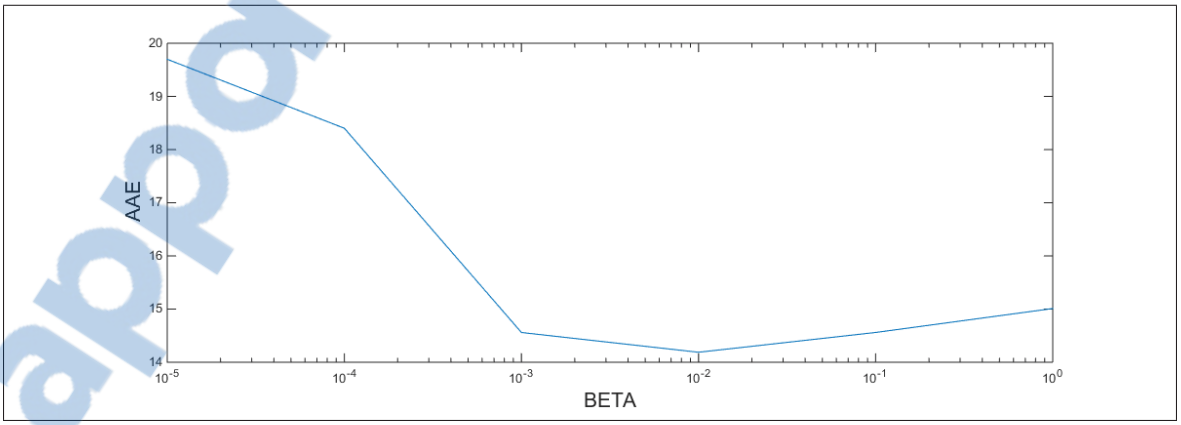

Figure 5.15 Comparaison de l'effet de la variation de β sur les SP (AAE)
maximum : 19.6987
minimum : 14.188

La variation de ce paramètre n'a pas d'effet notable sur le temps d'estimation du mouvement et le paramètre β n'affecte pas les calculs de la matrice de similitude du voisinage. Par contre,

il est possible de faire varier le paramètre qui détermine le nombre de pixels considéré par SP ce qui aura un impact sur les temps de calcul et sur les résultats obtenus. Donc en fixant le paramètre β à la meilleure valeur possible trouvée précédemment on obtient les résultats suivants. Avec la figure 5.16, on remarque que le temps d'exécution augmente de façon linéaire avec le nombre de pixels considéré par SP ce qui correspond avec l'analyse de l'algorithme. Les mesures d'erreur sont présentées dans les figures 5.17 et 5.18.

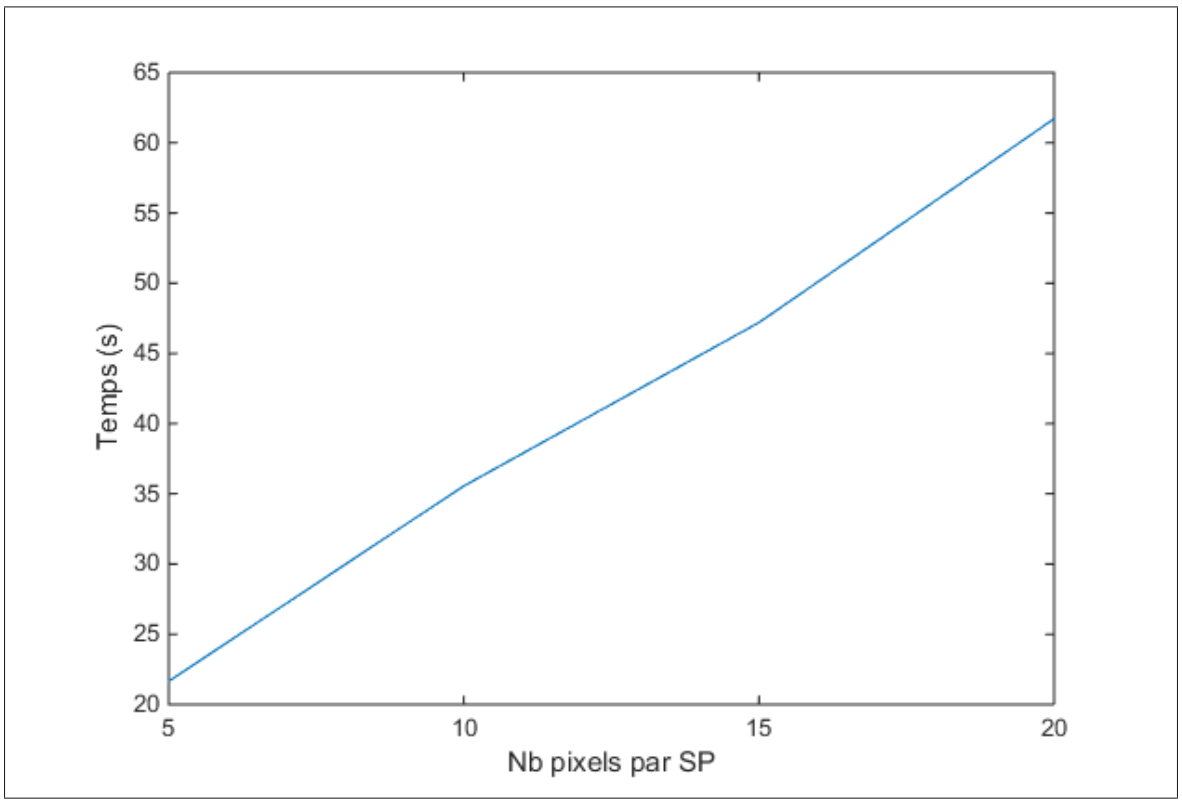

Figure 5.16 Comparaison de l'effet de la variation du nombre de pixels considérés par SP (Temps en secondes)

5.7 Comparatif avec d'autres méthodes

Dans le tableau comparatif 4.1, on compare les scores obtenus selon les deux métriques de comparaison proposées par Middlebury qui est la source des images. Les méthodes de **NNF-Local** et de **NNF-field** sont deux autres méthodes répertoriées sur le site comparatif des méthodes

Figure 5.17 Comparaison de l'effet de la variation du nombre de pixels considérés par SP (AEE)

et elles sont classées comme ayant obtenu parmi les meilleurs résultats sur l'image **Urban 2** (l'image Urban 3 n'est pas listée en tant que comparatif). Les valeurs calculées ci-dessous qui sont marquées comme arrondies sont obtenues en arrondissant au pixel près les valeurs de déplacements dans la référence proposée par la méthode pour tenter d'ignorer les erreurs dues aux problèmes d'arrondis. Dans les cas où la taille de l'image est diminuée, l'image de référence est aussi diminuée du même facteur et les vecteurs de mouvement sont aussi multipliés par le même facteur.

La comparaison avec les autres méthodes n'est pas présentée pour montrer que la méthode proposée est meilleure ou pas puisque pour effectuer la comparaison efficacement, il faudrait aller chercher des comparaisons sur toutes les images qui font partie du test. Ces valeurs sont seulement présentes pour donner une idée des valeurs que d'autres méthodes obtiennent sur ces deux métriques. La mesure d'angle moyen donne de moins bons résultats dans notre cas

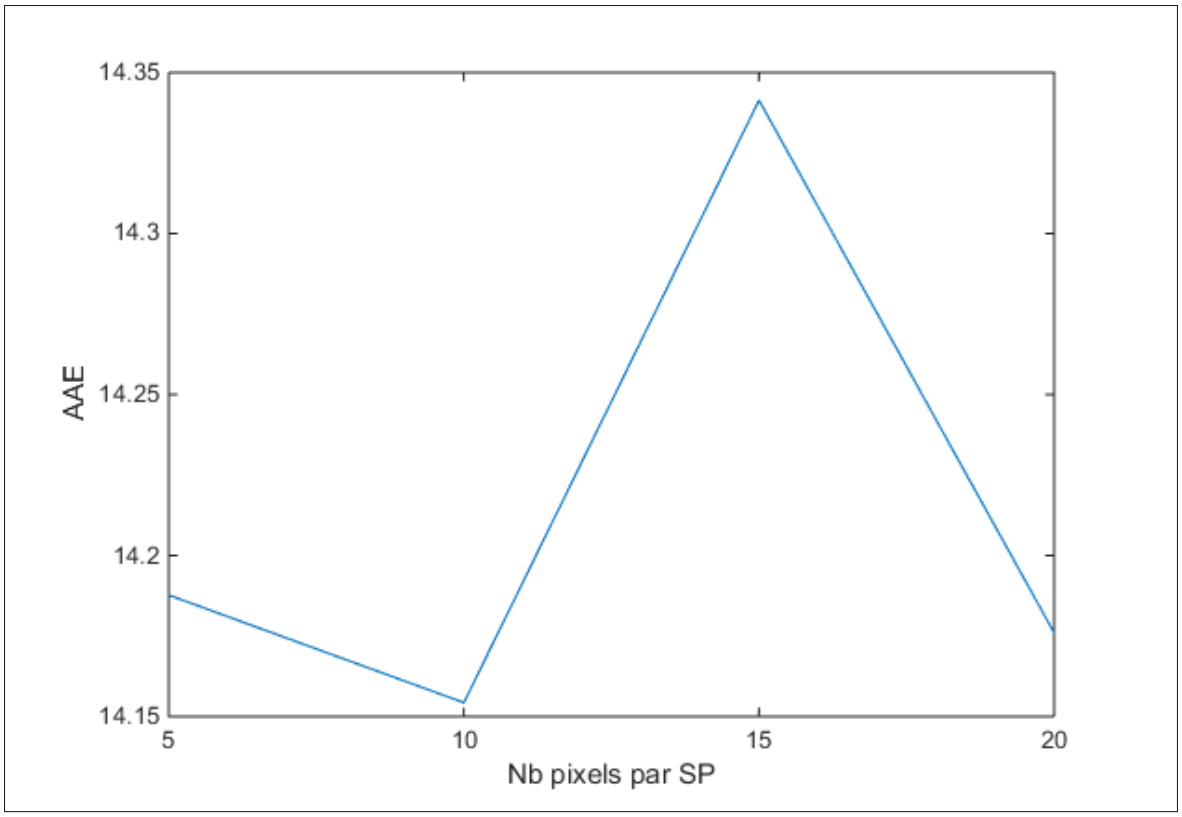

Figure 5.18 Comparaison de l'effet de la variation du nombre de pixels considérés par SP (AAE)

Tableau 5.1 Comparaison Urban avec autres méthodes

Méthode	Images	arrondi	AEE	AAE	Beta	Échelle
NNF-Local	**Urban**	-	0.23	2.39	-	1
NN-field Chen *et al.* (2013b)	**Urban**	-	0.52	1.89	-	1
Méthode proposée	**Urban2**	non	0.41	6.79	.0002	0.35
Méthode proposée	**Urban3**	non	0.31	6.43	0.00002	0.35
Méthode proposée	**Grove2**	non	0.321	11.0	0.001	0.35
Méthode proposée	**Grove2**	oui	0.58	5.57	0.001	0.35
Méthode proposée	**Grove3**	non	0.467	5.57	0.001	0.35
Méthode proposée	**Dimetrodon**	oui	0.225	9.247	0.01	0.35

en partie parce que la méthode proposée est discrète alors une petite erreur de distance peut se traduire par une erreur de plusieurs degrés si le centre du pixel résultant tombe à environ 1/2

pixel de distance. Il en résulte probablement une erreur angulaire moyenne un peu plus élevée qu'elle le serait si la méthode supportait des résultats continus.

5.7.1 Autres résultats

Certaines autres images analysées sont aussi intéressantes d'un point de vue technique. Certains de ces tests sont effectués sur des images réelles et non pas de synthèse ce qui parfois augmente le niveau de difficulté. Elles font partie de différents autres ensembles de tests d'images mais les résultats sont plus difficilement quantifiables puisque ces ensembles d'images n'incluent pas d'images de références pour permettre de quantifier les erreurs dans les résultats obtenus. Il faut donc évaluer les résultats obtenus de façon visuelle ou en appliquant un algorithme d'interpolation en utilisant la première image et le mouvement calculé pour tenter de reconstruire la deuxième image. Cependant, cela ne donne pas toujours des bons résultats puisque quand un objet se déplace devant un autre objet, il ne faisait pas partie de la portée de cette étude d'arriver à remplir le trou laissé par un objet déplacé artificiellement dans l'image. D'autres recherches se concentrent actuellement encore à résoudre ce problème. C'est pour cette raison que ces images sont étudiées de façon différente aux autres images présentées précédemment.

Parmi les exemples, il y a les images de Marple qui représente un homme étendant le bras pour aller décrocher un téléphone. La scène ne contient pas vraiment de mouvement autre que celui du personnage principal.

Le résultat de cette image n'a pas été coloré selon le mouvement trouvé par la méthode mais plutôt une couleur a été attribuée à chaque pixel en fonction de la similarité entre les *superpixels* et les *superpixels* voisins pour visualiser les régions plus générales de mouvement. Cette façon de présenter se rapproche plus de la segmentation par mouvement mais elle a été utilisée pour valider que la méthode avec les *superpixels* produisait des zones de mouvement qui respectaient bien les frontières réelles dans l'image. Comme on peut le voir dans cette image, le bras et le visage sont presque respectés à 100% et il faudrait probablement simplement régler

Figure 5.19 Images 20 et 21 de Marple

Figure 5.20 Résultats du mouvement Marple

l'algorithme de génération des superpixels pour s'assurer qu'il respecte bien les frontières dans l'image.

CHAPITRE 6

DISCUSSION SUR LA MÉTHODE - GÉNÉRATION DE CARTES DE PROFONDEUR

6.1 Méthodes semi-automatiques

Pour résoudre le problème de la génération de cartes de profondeur, une méthode semi-automatique semble être à favoriser. Tout au long des expérimentations et de l'analyse de la littérature sur le domaine, la structure et les caractéristiques de la scène à traiter semblent être une constante source de problèmes. D'ici à ce que les méthodes de calculs permettent d'analyser la scène préalablement au calcul de la solution, un humain est encore mieux placé pour pouvoir fournir des indices à la méthode sous la forme d'annotations pour minimiser l'impact de ces problèmes liés à la scène. Comme mentionné aussi plusieurs fois, cela permet un plus grand contrôle sur les résultats obtenus par la méthode et peut aussi permettre d'engager une itération d'amélioration avec l'utilisateur en lui permettant de corriger certains défauts dans la carte de profondeur générée.

C'est cette logique qui a guidé le choix du développement d'une méthode semi-automatique et même après les expérimentations, cela semble toujours l'option la plus logique lorsque l'occasion le permet. C'est pour cette raison qu'une partie des efforts ont été orientés vers l'amélioration de la considération des annotations dans la méthode ainsi qu'une tentative pour trouver la meilleure façon de permettre à l'utilisateur d'annoter les images. Évidemment que le but principal reste de sauver du temps pour la génération de cartes de profondeur par rapport à une méthode manuelle, alors ces annotations apportent chacune une grande amélioration du résultat obtenu. Il est envisageable qu'une méthode semi-automatique nécessite que l'utilisateur connaisse les particularités de la méthode pour savoir où placer les meilleures annotations.

Ayant obtenu de meilleurs résultats avec la méthode principale sans avoir un besoin aussi important de l'estimation de mouvement, la principale direction d'amélioration de la méthode aurait été la ou les façons de permettre aux utilisateurs d'annoter les images et de corriger les

résultats obtenus, autant au niveau de la mécanique d'annotation que de l'incorporation des annotations dans le modèle mathématique proposé.

6.2 Déplacement des pixels et propagation inter images

La génération d'une carte de profondeur d'une seule image avec des annotations d'un utilisateur n'est pas la difficulté principale du problème à résoudre. En fait, c'est beaucoup plus la génération des cartes de profondeur des images qui ne sont pas annotées par l'utilisateur qui apporte de la complexité au processus. L'information qui a été calculée à partir de la première image doit maintenant être propagée vers les autres images. Cela comprend les annotations et l'estimation originale de la profondeur. Cette propagation dépend très fortement de la capacité de la méthode à calculer le mouvement des pixels entre les images.

Une mauvaise correspondance trouvée entre des pixels pourrait causer de grandes erreurs sur le résultat obtenu. Et une erreur de calcul entre deux images aurait un fort impact sur le résultat global puisque l'information est propagée linéairement et que chaque image est influencée par la précédente dans la série. Il devient très difficile de sortir d'un cas où l'on serait en erreur puisque la méthode à l'étape de calculer une image va être fortement influencée par la précédente, les erreurs seront répliquées et souvent accentuées à travers la suite d'images.

Rapidement au travers des expérimentations plusieurs problèmes ont été rencontrés, qui ont été mentionnés précédemment : le manque d'information dans les régions non texturées, le problème des occlusions, mais surtout le problème de "ghosting" causé par une propagation inter images fautive qui venait d'une mauvaise estimation du mouvement. Les autres problèmes semblaient pouvoir être réglés en ajoutant certains calculs pour détecter les occlusions, par exemple, mais le problème d'une mauvaise estimation de mouvement nécessitait d'être résolu entièrement.

CHAPITRE 7

DISCUSSION SUR LA MÉTHODE - ESTIMATION DE MOUVEMENT

7.1 Améliorations et pistes de solutions aux problèmes rencontrés

7.1.1 Importance du descripteur

Tout au long des expérimentations, il est ressorti une forte tendance que la qualité des résultats de la méthode dépendait fortement de la qualité du descripteur. Le descripteur est un vecteur de caractéristiques extraites d'un pixel ou d'une zone de pixel qui sert à effectuer la comparaison entre deux pixels ou zones. Dans la littérature, on retrouve toutes sortes de descripteurs de pixels qui s'adaptent souvent à l'application qui l'utilise. Dans notre cas, un descripteur ne devait pas être trop complexe car le plus d'information contenue dans le descripteur, plus longue sera la comparaison entre deux pixels. Dans notre étude, comme le temps de calcul restait un critère pour la performance de la méthode, il fallait se limiter à un descripteur qui permettait de régler certains problèmes de la méthode tout en restant relativement simple.

Après expérimentations, en plus d'un descripteur de la couleur de chaque pixel, un descripteur de voisinage a été choisi pour plusieurs raisons. Comme la méthode cherche le mouvement de manière dense, il faut donc décrire chaque pixel. Cela peut entraîner plusieurs problèmes qui sont reliés à une absence d'information quand un pixel est dans une zone sans texture. En effet, peu importe la qualité d'un descripteur ne tirant son information que du pixel même, si ce pixel est au milieu d'une large zone où tous les pixels sont semblables, il n'y a pas de moyen pour le différencier des pixels voisins. C'est pour cette raison qu'un descripteur considérant le voisinage dans une certaine fenêtre a été choisi. En considérant les voisins du pixel, on se rapproche d'une comparaison de "textures" ce qui améliore grandement la performance dans les zones faiblement texturées. Cela apporte aussi des améliorations aussi dans les zones où les textures sont répétitives puisque, dépendant de la grandeur de la fenêtre du descripteur, il est probable qu'au moins quelques pixels vont considérer l'extérieur de cette zone répétitive et

influencer les pixels à l'intérieur de la zone qui manqueraient d'information. Dans notre cas, ces problèmes sont diminués puisque le descripteur utilisé considère les voisins (RVB) ainsi que les gradients de l'image.

Avant d'arriver à ce descripteur, plusieurs autres ont été testés pour vérifier leur efficacité. Plusieurs de ceux-ci sont proposés dans différentes autres méthodes et ont été évalués pour valider leur solution apportée aux problèmes fréquents mentionnés précédemment. Par exemple, dans la littérature, on peut retrouver le *Local Binary Pattern* et un descripteur qui ressemble à celui utilisé, l'*Histogram of Gradients*. Les principes de ces descripteurs suivent aussi une tendance de décrire la texture autour d'un pixel au lieu de seulement prendre de l'information sur le pixel lui-même. Certains de ces descripteurs ne présentaient tout simplement pas l'information recherchée pour régler les problèmes rencontrés. Ces descripteurs ont été étudiés après avoir déjà mis en place le descripteur utilisé donc il y avait déjà une base pour permettre une comparaison entre les résultats obtenus avec le descripteur déjà choisi et les autres options.

Dans le cas du *LBP*, on sélectionne une fenêtre autour d'un pixel, on prend les valeurs de ces pixels de façon circulaire et pour chaque valeur, si elle est plus grande que celle du pixel central on attribue un 1 à ce pixel, sinon un 0. Cela nous donne donc un chiffre sur 8 bits par pixel dans la fenêtre et on calcule par la suite l'histogramme de fréquence de ces chiffres sur toute la fenêtre. Cet histogramme peut être par la suite traduit en un descripteur de longueur 256 qui contient la fréquence de chaque possibilité de valeur sur 8 bits. L'intention de ce descripteur est d'obtenir de l'information sur les voisins du pixel tout en s'assurant que cette information n'est pas dépendante de la rotation des objets dans l'image. Ce descripteur ajoutait un peu de complexité durant le calcul de la solution et au final ne semblait pas offrir d'amélioration sur les résultats obtenus. Cependant, par sa composition, il pourrait être adapté pour les SP puisqu'il fonctionne déjà selon un principe de description par fenêtre.

L'importance d'un bon descripteur est confirmée par l'analyse de la littérature ainsi que par les expérimentations effectuées. Plusieurs des problèmes que la méthode avait au début des expérimentations ont été réglés suite à l'implémentation d'un descripteur de texture.

7.1.2 Apprentissage des paramètres de la méthode

La méthode s'appuie sur plusieurs paramètres pour produire les résultats et la majorité des paramètres ont été déterminés empiriquement à travers les expérimentations. Cependant, il serait intéressant, étant donné que certaines des séries d'images fournissent des images de références pour calculer des métriques d'erreur, de trouver un processus d'apprentissage des différentes valeurs des paramètres donnant les meilleurs résultats pour ces images. Certains essais ont été réalisés dans cette direction au début de l'expérimentation avec la méthode mais une fois la complexité augmentée par le descripteur contenant un nombre élevé de valeurs, les techniques d'apprentissage itératif de base sont devenues rapidement trop lentes pour couvrir toutes les combinaisons possibles des différents poids de chaque élément dans le descripteur.

La meilleure solution serait d'établir les paramètres par défaut de la méthode à partir de ceux déterminés empiriquement qui, en moyenne, donnent les meilleurs résultats avec la méthode. À partir de cette base, une méthode itérative serait mise en place pour déterminer les paramètres optimaux pour une image spécifique. Les poids qui seraient le plus intéressants à calculer avec une technique comme celle-ci seraient la valeur accordée à chaque portion du descripteur. Après plusieurs itérations, il deviendrait clair si le fait d'ajouter, par exemple, le gradient des pixels voisins est utile à améliorer la méthode ou si le poids attribué expérimentalement tend vers 0.

Cette façon de procéder permettrait aussi de tester si l'ajout de certains autres descripteurs dans la formule améliore les résultats ou si, comme dans le cas des gradients, on finit par trouver que la meilleure valeur est de 0. Logiquement, on devrait aussi observer que tout dépendant du contenu des images, certaines portions du descripteur seraient plus ou moins utiles. Probablement que, dans le cas de petits objets très différents de leurs alentours, les gradients et le voisinage éloigné sont peu utiles et peuvent même nuire aux résultats puisque cela ajoute un niveau de complexité inutile qui a un impact négatif sur la certitude des déplacements trouvés. Cela dépasse les objectifs de l'élaboration de la méthode, mais une fois celle-ci perfectionnée pour au moins une image, des itérations avec un utilisateur seraient intéressantes. Par exemple,

dans le cas où une méthode interactive serait prévue, l'utilisateur pourrait visualiser le résultat obtenu et modifier les paramètres pour recalculer les résultats en fonction de ce qu'il pense améliorerait le résultat final. L'utilisateur devrait d'abord bien comprendre ce que la modification de chaque paramètre entraîne mais la méthode devrait aussi lui fournir un bon point de départ pour commencer.

7.1.3 Considération des occlusions

Un des problèmes importants de la méthode est qu'il n'y a pas de traitement ou de technique permettant de détecter et de tenir compte des occlusions. Pour référence, les occlusions sont des pixels qui ne sont pas présents dans les deux images parce qu'ils sont occlus par un objet généralement passant devant ceux-ci à travers la séquence d'images. Ces pixels posent un problème puisque la méthode va essayer de suivre le déplacement de chaque pixel d'une image à l'autre. Dans un cas général, les séquences d'images comptent peu de ces pixels d'une image à l'autre mais il est bon d'en tenir compte. Pour pouvoir en tenir compte, il faut en premier arriver à les détecter. Il existe plusieurs méthodes dans la littérature pour arriver à détecter les occlusions et une technique générale peut être appliquée à la méthode proposée.

En partant de l'affirmation que la méthode donne des bons résultats et que les pixels en occlusions sont des pixels qui ne sont pas présents dans une des deux images. On peut planifier un processus itératif où la méthode est appliquée une fois dans l'ordre chronologique des images et une fois dans l'ordre inverse. Ensuite, en prenant les mouvements trouvés pour chaque pixel et en faisant correspondre les pixels un à un, on peut assumer que les pixels n'ayant pas des mouvements opposés sont des bons candidats comme étant des pixels en occlusion. Il en va de même pour les pixels ayant plusieurs candidats très similaires dans un sens et plusieurs pixels ayant le même pixel comme mouvement le plus probablement dans le sens contraire.

Les deux figures suivantes illustrent comment la détection des occlusions peut procéder. Dans la première figure, il n'y aura pas d'occlusion de détectée puisque chaque pixel trouve son

pixel correspondant dans l'image suivante. Dans la deuxième figure, les pixels bleus sont des candidats pour des occlusions puisque la correspondance entre les pixels n'est pas du 1 pour 1.

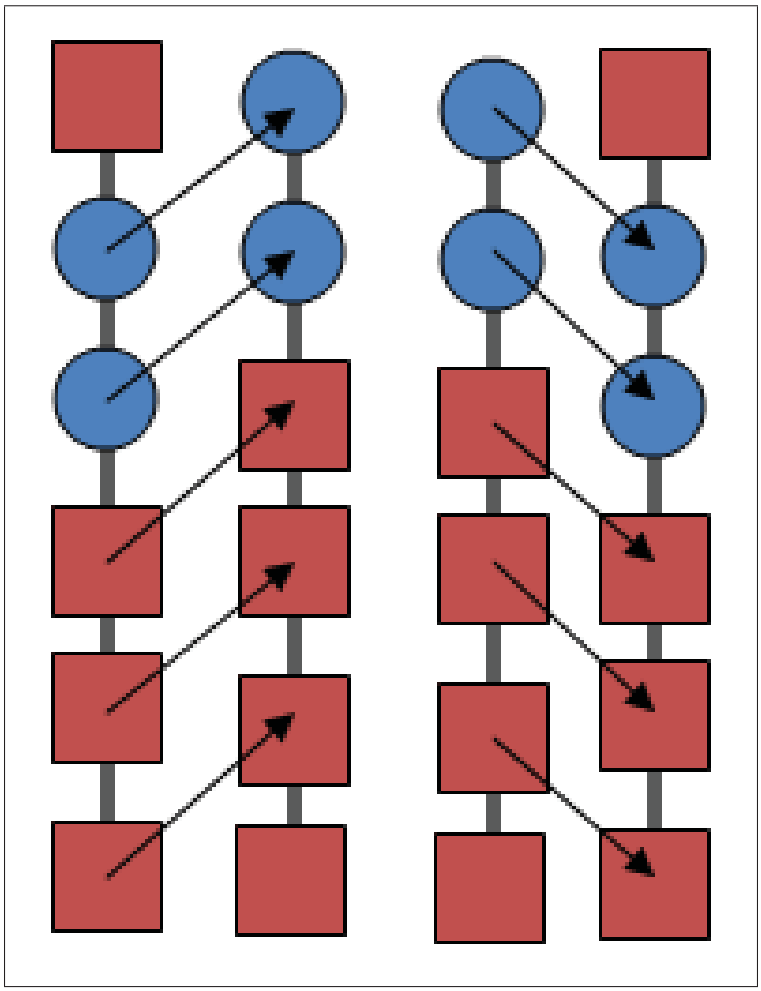

Figure 7.1 Mouvements sans occlusions (Image1->Image2 et Image2->Image1)

La détection des occlusions est seulement la première étape pour arriver à en tenir compte. Il faut par la suite trouver une technique pour assigner à un pixel une valeur qui signifie que le pixel est en occlusion comme vecteur de mouvement. Par exemple, ce résultat peut être obtenu en tenant compte du mouvement des autres pixels aux alentours qui ne sont pas en occlusion. Dans certains cas, c'est justement les pixels voisins qui vont recouvrir certains pixels pour les occlure dans l'image suivante. Dans ces cas, les pixels devraient tenir compte des frontières

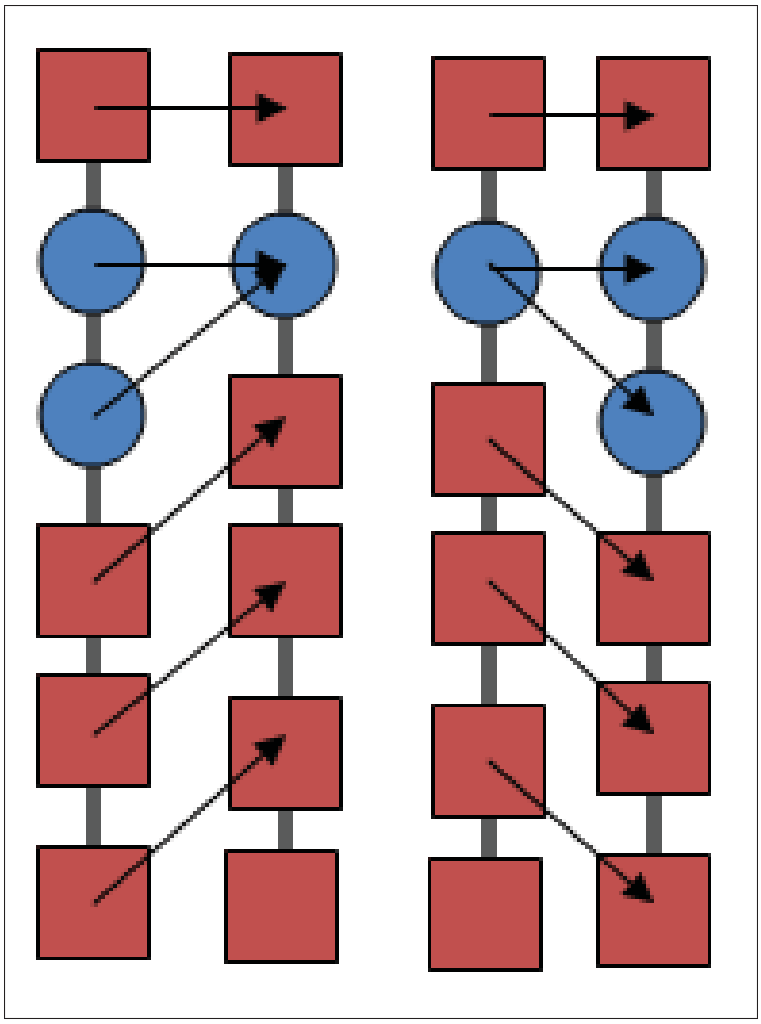

Figure 7.2 Mouvements avec occlusions potentielles
(Image1->Image2 et Image2->Image1)

originelles de l'objet maintenant occlus pour décider du mouvement le plus probable. Dans le cas de la méthode proposée, les pixels aux frontières des objets qui deviennent occlus vont "pousser" les pixels voisins similaires comme si l'objet au complet se déplaçait plus qu'il ne se déplace réellement. Il faudrait donc trouver une technique qui permet de tenir compte des occlusions et de réguler ces erreurs de mouvements qui sont générées.

CHAPITRE 8

CONCLUSION

Cette étude a servi à déterminer le potentiel d'une méthode d'optimisation de contraintes sur l'estimation de la profondeur semi-automatique. Le but de la recherche étant de trouver une technique rapide d'annotation et de propagation d'information de la profondeur sur une séquence d'images, une technique basée sur un indice de confiance et le *Random Walker* a été développée. La technique consiste en 2 étapes, la première étant l'annotation d'une ou de plusieurs images clés dans la séquence, la seconde étant un processus itératif en deux phases en premier d'estimation de la profondeur des images clés et en deuxième de la propagation de l'information sur toute la séquence. Durant le développement de la méthode, des tests ont été effectués sur diverses images tout d'abord simples pour valider l'efficacité de la méthode d'estimation de la profondeur à partir des annotations et puis finalement sur des séquences d'images pour valider l'efficacité de la propagation de l'information. Les conclusions tirées des expérimentations sont que la méthode d'estimation de la profondeur est performante et résistante au bruit même avec un petit nombre d'annotations, cependant, la méthode de propagation de l'information a rencontré plusieurs problèmes communs à ce genre de méthodes. Par la suite, la recherche s'est orientée vers trouver une méthode d'estimation de mouvement dense qui permettrait d'obtenir une meilleure idée du déplacement des pixels entre deux images consécutives.

Il est important pour la méthode de trouver la correspondance entre deux pixels de deux images consécutives puisque la propagation de l'information dépend de celle-ci. Avec une mauvaise correspondance, plusieurs problèmes se glissent dans les résultats. La méthode recherchée pour l'estimation de mouvement doit aussi être rapide et automatique pour être utilisable dans celle d'estimation de profondeur. La méthode développée est probabiliste c'est-à-dire qu'elle va trouver la probabilité, pour chaque pixel, de se déplacer dans toutes les directions définies dans le *kernel* et, pour trouver le résultat final, la probabilité maximale est choisie. Une autre particularité est que les probabilités des pixels dans l'image peuvent influencer les pixels avoisinants

soit pour renforcer la confiance soit pour permettre de distinguer entre deux options également probables. La méthode développée a été testée sur un ensemble d'images publiées par *Middlebury* pour quantifier la performance de méthodes d'estimation du mouvement en calculant deux mesures d'erreurs, l'erreur de déplacement moyen et l'erreur angulaire moyenne. Les tests ont été effectués sur deux des images proposées par *Middlebury* et les mesures d'erreurs obtenues sont bien situées comparées aux autres méthodes listées. Cependant, le processus complet de classification de la performance de la méthode n'a pas été effectué puisque dans les conditions établies par *Middlebury*, la méthode doit fournir les meilleurs résultats sur toutes les images proposées sans changement dans des paramètres de contrôle alors que la méthode développée est construite pour permettre d'ajuster les résultats obtenus avec des paramètres de contrôle.

Même si la performance de la méthode se classe bien par rapport à d'autres méthodes proposées selon les mesures d'erreur, les résultats obtenus visuellement présentent parfois des zones concentrées de fortes erreurs qui n'ont pas beaucoup d'impact dans une mesure moyenne, mais qui pourraient avoir un impact important si les résultats étaient utilisés pour la propagation de l'information. Certains problèmes communément rencontrés par des techniques d'estimation de mouvement sont aussi présents dans la méthode proposée comme la détection et le traitement d'occlusions, les zones où la différence entre les pixels est tellement minime que l'on n'arrive pas à différencier un pixel d'un autre (*texture-less regions*) et finalement des zones contenant des textures fortement répétitives qui, encore une fois, rendent la distinction entre deux pixels difficile à faire. Des pistes de solutions pour régler ces problèmes sont proposées dans ce document au chapitre 5. D'autres améliorations à la méthode ont aussi été proposées pour créer un paramètre de contrôle d'un compromis entre la rapidité de calcul et la précision des résultats obtenus avec des *Superpixels*. La possibilité d'utiliser d'autres techniques pour déterminer la stéréo-correspondance a aussi été ajoutée et testée avec l'algorithme SIFT, cet ajout permet aussi de considérer des annotations éventuelles comme étant des probabilités absolues.

Pour finir, la recherche présentée dans ce document, même si elle n'a pas abouti à un produit final d'estimation semi-automatique de la profondeur prêt à être utilisé dans le contexte de conversion d'images 2D à 3D, aura permis de tester la viabilité d'une méthode probabiliste pour

accomplir cette tâche. Elle aura aussi permis de donner une bonne idée du temps qui pourrait être sauvé pour la conversion comparativement à une méthode manuelle. La viabilité de la technique de *Random Walker* ne semblait pas non plus avoir été explorée dans la littérature pour résoudre le problème d'estimation de la profondeur comme utilisé dans la méthode proposée. Finalement, avec les pistes de solution proposées pour les problèmes rencontrés il y a espoir d'améliorer grandement les résultats obtenus par la méthode dans les situations problématiques énoncées précédemment.

BIBLIOGRAPHIE

Baker, S., D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, And R. Szeliski. 2011. « A database and evaluation methodology for optical flow ». *International Journal of Computer Vision*, vol. 92, n° 1, p. 1–31.

Black, M. J. And P. Anandan. 1996. « The Robust Estimation of Multiple Motions-Parametric and Piecewise smooth flow fields ». *Computer Vision and Image Understanding*, vol. 63, n° 1, p. 75–104.

Bolecek, L. And V. Ricny. jul 2013. « The estimation of a depth map using spatial continuity and edges ». In *2013 36th International Conference on Telecommunications and Signal Processing (TSP)*. p. 890–894. IEEE.

Chalidabhongse, J. And C. C. Jay Kuo. 1997. « Fast motion vector estimation using multiresolution-spatio-temporal correlations ». *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, n° 3, p. 477–488.

Chan, S. H., D. T. Vo, And T. Q. Nguyen. 2010. « Subpixel motion estimation without interpolation ». In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. p. 722–725. IEEE.

Chen, M. J., L. G. Chen, T. D. Chiueh, And Y. P. Lee. 1995. « A New Block-Matching Criterion for Motion Estimation and its Implementation ». *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, n° 3, p. 231–236.

Chen, Z., H. Jin, Z. Lin, S. Cohen, And Y. Wu. jun 2013a. « Large displacement optical flow from nearest neighbor fields ». In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. p. 2443–2450. IEEE.

Chen, Z., H. Jin, Z. Lin, S. Cohen, And Y. Wu. jun 2013b. « Large Displacement Optical Flow from Nearest Neighbor Fields ». In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. p. 2443–2450. IEEE.

Cheng, C.-M., X.-A. Hsu, And S.-H. Lai. jul 2010. « A novel structure-from-motion strategy for refining depth map estimation and multi-view synthesis in 3DTV ». In *2010 IEEE International Conference on Multimedia and Expo*. p. 944–949. IEEE.

Cigla, C. And A. A. Alatan. may 2009. « Temporally consistent dense depth map estimation via Belief Propagation ». In *2009 3DTV Conference : The True Vision - Capture, Transmission and Display of 3D Video*. p. 1–4. IEEE.

Deng, X. And X. Jiang. dec 2008. « Automatic depth map estimation of monocular indoor environments ». *... Technology, 2008. MMIT' ...*, p. 646–649.

Fleet, D. J. And Y. Weiss. 2005. « Optical flow Estimation ». *Handbook of Mathematical Models*, vol. 19, p. 239–258.

Freiling, B. And T. Schumann. 2013. « Block based depth map estimation algorithm for 2D-to-3D conversion ». … *Electronics (ISCE), 2013* …, p. 53–54.

Grady, L. Nov 2006. « Random Walks for Image Segmentation ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, n° 11, p. 1768-1783.

Han, K. And K. Hong. jan 2011. « Geometric and texture cue based depth-map estimation for 2D to 3D image conversion ». … *ICCE), 2011 IEEE International Conference on*, p. 651–652.

Harrison, A. And D. Joseph. dec 2012. « Maximum likelihood estimation of depth maps using photometric stereo ». *Pattern Analysis and Machine* …, vol. 34, n° 7, p. 1368–1380.

Huang, Y. W., C. Y. Chen, C. H. Tsai, C. F. Shen, And L. G. Chen. 2006. « Survey on block matching motion estimation algorithms and architectures with new results ». *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 42, n° 3, p. 297–320.

Jain, J. 1981. « Displacement measurement and its application in interframe image coding ». *IEEE Transactions on Communications*, vol. 29, p. 1799–1808.

Jung, J.-i. And Y.-s. Ho. jun 2010. « Depth map estimation from single-view image using object classification based on Bayesian learning ». In *2010 3DTV-Conference : The True Vision - Capture, Transmission and Display of 3D Video*. p. 1–4. IEEE.

Knorr, S. And T. Sikora. 2006. « An image-based rendering (IBR) approach for realistic stereo view synthesis of tv broadcast based on structure from motion ». *Proceedings - International Conference on Image Processing, ICIP*, vol. 6.

Koga, T., K. Iinuma, A. Hirano, Y. Iijima, And T. Ishiguro. 1981. « Motion compensated interframe coding for video conferencing ». *Proc. Nat. Telecommun. Conf., New Orleans*, p. G5.3.1–5.3.5.

Konrad, J., M. Wang, P. Ishwar, C. Wu, And D. Mukherjee. 2013. « Learning-based, automatic 2D-to-3D image and video conversion ». *IEEE Transactions on Image Processing*, vol. 22, n° 9, p. 3485–3496.

Lee, J., S. Yoo, C. Kim, And B. Vasudev. jun 2013. « Estimating Scene-Oriented Pseudo Depth With Pictorial Depth Cues ». *IEEE Transactions on Broadcasting*, vol. 59, n° 2, p. 238–250.

Lee, S.-B., K.-J. Oh, And Y.-S. Ho. may 2008. « Segment-Based Multi-View Depth Map Estimation Using Belief Propagation from Dense Multi-View Video ». In *2008 3DTV Conference : The True Vision - Capture, Transmission and Display of 3D Video*. p. 193–196. IEEE.

Lee, S., J. Lee, M. H. Hayes, A. K. Katsaggelos, And J. Paik. mar 2012. « Single camera-based full depth map estimation using color shifting property of a multiple color-filter aperture ». In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. p. 801–804. IEEE.

Li, W. And E. Salari. 1995. « Successive Elimination Algorithm for Motion Estimation ». *IEEE Transactions on Image Processing*, vol. 4, n° 1, p. 105–107.

Lin, Y.-H., T.-W. Chiang, T. Tsai, And M.-J. Hsiao. nov 2010. « Fast 3D image depth map estimation using wavelet analysis ». In *2010 International Conference on Audio, Language and Image Processing*. p. 1160–1163. Dept. of Multimedia Design, Chihlee Institute of Technology : IEEE.

Liu, C. And L. Christopher. may 2012. « Depth map estimation from motion for 2D to 3D conversion ». *Electro/Information Technology (EIT), . . .*, vol. 1, p. 1–4.

Lowe, D. G. 1999. « Object recognition from local scale-invariant features ». *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, n° [8, p. 1150–1157.

Malik, A., S. Shim, And T. Choi. 2007. « Depth map estimation using a robust focus measure ». *Image Processing, 2007. ICIP . . .*, p. VI – 564–VI – 567.

Mendapara, P. jul 2010. « An efficient depth map estimation technique using complex wavelets ». *Multimedia and Expo (. . .*, p. 1409–1414.

Mendapara, P., R. Minhas, And Q. J. Wu. oct 2009. « Depth map estimation using exponentially decaying focus measure based on SUSAN operator ». In *2009 IEEE International Conference on Systems, Man and Cybernetics*. p. 3705–3708. IEEE.

Mileva, Y., A. Bruhn, And J. Weickert. 2007. « Illumination-Robust Variational Optical Flow with Photometric Invariants ». *Dagm*, p. 152–162.

Phan, R., R. Rzeszutek, And D. Androutsos. sep 2011. « Semi-automatic 2D to 3D image conversion using scale-space Random Walks and a graph cuts based depth prior ». In *2011 18th IEEE International Conference on Image Processing*. p. 865–868. IEEE.

Po, L. M., W. C. Ma, Lai-Man Po, And Wing-Chung Ma. 1996. « A novel four-step search algorithm for fast block motion estimation ». *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, n° 3, p. 313–317.

Rajan, D. And S. Chaudhuri. sep 2003. « Simultaneous estimation of super-resolved scene and depth map from low resolution defocused observations ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, n° 9, p. 1102–1117.

Saxena, A., M. Sun, And A. Y. Ng. 2009. « Learning 3D scene structure from a single still image. ». *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, n° 5, p. 824–40.

Simoncelli, E., E. Adelson, And D. Heeger. 1991. « Probability distributions of optical flow ». In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. p. 310–315. IEEE Comput. Sco. Press.

Tam, W. J., C. Vazquez, And F. Speranza. 2009. « Three-dimensional TV : a novel method for generating surrogate depth maps using colour information ». *Proceedings of SPIE*, vol. 7237, p. 72371A–72371A–9.

Tham, J. Y., S. Ranganath, M. Ranganath, And A. A. Kassim. 1998. « A novel unrestricted center-biased diamond search algorithm for block motion estimation ». *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, n° 4, p. 369–377.

Tzovaras, D., M. G. Strintzis, And H. Sahinoglou. 1994. « Evaluation of multiresolution block matching techniques for motion and disparity estimation ». *Signal Processing : Image Communication*, vol. 6, n° 1, p. 59–67.

Vazquez, C., L. Zhang, F. Speranza, N. Plath, And S. Knorr. apr 2013. 2D-to-3D Video Conversion : Overview and Perspectives. Dufaux, F., Béatrice Pesquet-Popescu, And Marco Cagnazzo, editors, *Emerging Technologies for 3D Video : Creation, Coding, Transmission and Rendering*, p. 37–61. John Wiley & Sons, Ltd, Chichester, UK. ISBN 9781118355114. doi : 10.1002/9781118583593.ch3. <http://doi.wiley.com/10.1002/9781118583593http://doi.wiley.com/10.1002/9781118583593.ch3>.

Ward, B., Sing Bing Kang, And E. P. Bennett. jan 2011. « Depth Director : A System for Adding Depth to Movies ». *IEEE Computer Graphics and Applications*, vol. 31, n° 1, p. 36–48.

Xu, X., L.-M. Po, K.-W. Cheung, K.-H. Ng, K.-M. Wong, And C.-W. Ting. aug 2012. « Watershed and Random Walks based depth estimation for semi-automatic 2D to 3D image conversion ». In *2012 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2012)*. p. 84–87. IEEE.

Yao, S.-J., L.-H. Wang, D.-X. Li, And M. Zhang. jul 2013. « A Real-Time Full HD 2D-to-3D Video Conversion System Based on FPGA ». In *2013 Seventh International Conference on Image and Graphics*. p. 774–778. IEEE.

Zhang, G., J. Jia, T. T. Wong, And H. Bao. 2009. « Consistent depth maps recovery from a video sequence ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, n° 6, p. 974–988.