TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AE	Erreur angulaire *angular error*
BPT	Arbre de partition binaire *binary partition tree*
BVSD	Base de données de Berkeley pour la segmentation vidéo *Berkeley video segmentation dataset*
CD	Détection consistante *consistent detection*
CMU	Base de données de Carnegie Mellon *Carnegie Mellon dataset*
CNN	Réseaux de neurones convolutionel *convolutional neural networks*
DAG	Graphe acyclique de profondeur *depth acyclic graph*
DIBR	Rendu basé image de profondeur *depth-image based rendering*
EMD	Distance earth mover *earth mover's distance*
EPE	Erreur du point d'arrivée *end point error*
F	Mesure-F *F-mesure*
FD	Fausse détection *false detection*
FN	Fausses négatives *false negatives*

FP	Fausses positives *false positives*
HOG	Histogrammes des gradients orientés *histograms of oriented gradients*
ID	Détection inconsistante *Inconsistent detection*
LA	Affine localement-pondérée *locally-weighted affine*
LBP	Patrons binaires locaux *local binary patterns*
LDOF	Flot optique de large déplacement *large displacement optical flow*
MD	Détection manquée *missed detection*
NW	Nadaraya-Watson *Nadaraya-Watson*
ORI	Indice supérieur à l'aléatoire *over random index*
P	Précision *precison*
R	Rappel *recall*
RAG	Graphe de régions d'adjacence *region adjacency graph*
RGB	Rouge-vert-bleu *red-green-blue*
SED	Détecteur de contours structuré *structured edge detector*

SFM Structure déduite du mouvement
structure from motion

SIFT Caractéristiques invariantes aux transformations d'échelle
scale-invariant feature transform

SLIC Simple regroupement linéaire itératif
simple linear iterative clustering

SURF Caractéristiques robustes accélérées
speeded-up robust features

SVM Machine à vecteurs de support
support vector machine

TN Vraie négative
true negative

TP Vraies positives
true positives

INTRODUCTION

Au début des années 2000, les téléviseurs 3D ont fait leur apparition sur le marché grand public. Cette nouveauté a généré un véritable engouement à son lancement. Néanmoins, cette technologie, qui se voulait très prometteuse, n'a pas eu tout le succès escompté. Un des plus importants obstacles rencontrés est le manque de contenu 3D.

Plus récemment, avec l'arrivée des casques de réalité virtuelle sur le marché grand publique, le besoin en contenu 3D devient encore plus accru.

Le manque de contenu provoque un manque d'intérêt pour la 3D, ce qui a pour effet de ralentir cette industrie et ce ralentissement à son tour n'encourage pas la création de contenu 3D. Ces technologies risquent ainsi de se retrouver prises dans ce cercle sans fin.

Il existe aujourd'hui plusieurs dispositifs permettant l'acquisition de contenu 3D. Il s'agit de caméras stéréoscopiques ou des dispositifs comprenant des capteurs de profondeurs. Cependant, la quantité de contenu produite par ces procédés d'acquisition reste négligeable comparée à celle de contenu 2D traditionnel et ne permet pas de combler le manque de contenu 3D. En effet, ces nouveautés technologiques sont plus coûteuses, moins faciles à manipuler et surtout bien moins répandues que les caméras 2D monoscopiques. Et quand bien même ces dispositifs deviendraient plus abordables et répandus, ils ne permettraient pas de capturer à nouveau toutes les vidéos 2D déjà acquises.

Une solution possible pour atténuer la carence de contenu 3D est d'en générer à partir de contenu 2D ordinaire. Cette conversion s'effectue par la génération d'une carte de profondeur de chaque trame de la vidéo à convertir. Il est possible d'effectuer cette conversion manuellement, mais ce travail manuel artistique est lent et coûteux. À titre d'exemple, la conversion du film Titanic en 3D a nécessité soixante semaines de travail effectué par 300 infographistes, pour un coût total de 18 millions de dollars [1]. Si l'industrie du cinéma peut parfois se permettre ces coûts et délais de production élevés, ce n'est pas le cas pour la télévision. Ainsi,

1. http://lexpansion.lexpress.fr/high-tech/convertir-titanic-en-3d-un-travail-de-titan_1448417.html consutlé le 04-05-2017

beaucoup de recherches ont été menées pour rendre cette conversion abordable et plus rapide. Ces recherches visent soit à minimiser l'intervention humaine avec des méthodes dites semi-automatiques (Guttmann *et al.*, 2009; Cao *et al.*, 2011) soit à l'éliminer complètement (Wei, 2005; Zhang *et al.*, 2011) par des méthodes automatiques.

Pour effectuer la conversion, les méthodes automatiques exploitent les indices de profondeurs présents dans le contenu 2D. Ces indices peuvent être purement spatiaux ; ils traitent chaque trame d'une séquence indépendamment. Ce sont des indices statiques, comme les perspectives linéaires, les informations de couleur et de texture. D'autres indices utilisent les composantes spatiales et temporelles d'une vidéo. Ce sont des indices dynamiques, comme le parallaxe de mouvement.

Parmi les indices de profondeur, il en existe un qui peut appartenir à l'une ou l'autre des deux catégories selon la manière dont il est calculé : Il s'agit de l'indice de l'interposition des objets qui génère des occlusions dans les images. En effet, ces occlusions peuvent être déduites d'une image fixe par détection des jonctions en T, ou bien par exploitation du mouvement entre deux ou plusieurs trames. Contrairement à d'autres indices de profondeur qui ne peuvent être exploités que dans certains cas, tels que la brume atmosphérique présente uniquement en paysages extérieurs, les occlusions ont l'avantage d'être présentes dans tous les types de scènes. De plus, les occlusions sont robustes. En effet, si deux objets sont présents dans une scène et que les occlusions permettent de déduire qu'un des deux cache une partie de l'autre, il ne fait pas de doute que l'objet partiellement caché se trouve en arrière de celui qui le cache.

Il est vrai que les occlusions ne permettent pas de déduire un ordre absolu, mais seulement un ordre relatif. Cependant, l'ordre relatif peut suffire à générer du 3D. Selon des études cognitives (Koenderink *et al.*, 2001), le système visuel humains est plus conscient de l'ordre de profondeur que de la valeur absolue de la profondeur.

Parce que les occlusions déduites du mouvement sont plus fiables que celles déduites de manière statique, c'est à elles que cette étude s'intéresse. Dans se contexte, l'approche et les

résultats des travaux Palou (2013) et Salembier & Palou (2014) sont très intéressants. Les informations de couleur et de mouvement sont toutes deux utilisées pour obtenir une segmentation de la trame ainsi qu'un ordre relatif des objets de cette segmentation. Seulement, une des étapes de cette approche nécessite le calcul d'une estimation de mouvement par région selon un modèle quadratique. L'approche de ce présent travail vise à éliminer cette opération coûteuse sans perte de performance. Pour y arriver, cette approche prend avantage d'une méthode de calcul du flot optique qui préserve les contours (Revaud *et al.*, 2015), auxquelles des modifications ont permis de rendre les flots optiques en avant et en arrière cohérents. Ainsi, en ayant un flot optique préservant les frontières et cohérent avant-arrière, les occlusions peuvent être plus simplement calculées. Ces occlusions permettent à leurs tour de déduire l'ordre des objets. Les objets étant obtenus par une segmentation basée sur la construction d'un arbre binaire de partition et prenant compte les informations de couleur et de mouvement.

Le premier chapitre de ce mémoire présente une revue de littérature des travaux portant sur la conversion 2D à 3D. Le second chapitre détaille la solution explorée. Le troisième chapitre comporte les résultats obtenus ainsi qu'une discussion de ces résultats. Les perspectives possibles à ce travail sont présentées à la conclusion générale de ce manuscrit.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

Ce premier chapitre se compose de six sections. La première expose la différence entre contenu 2D et contenu 3D. Elle est suivie d'une revue de la littérature des approches qui effectuent la conversion 2D à 3D. Puis, les approches basées sur les occlusions sont présentées. La quatrième section traite du calcul du flot optique, outil souvent utilisé dans la conversion 2D à 3D. Quant à la cinquième section, elle détaille une méthode de conversion 2D à 3D automatique qui repose sur la détection des occlusions. Une conclusion constitue la dernière section du chapitre.

1.1 Différences entre contenu 2D et contenu 3D

Supposons que nous disposons d'une vidéo 2D, quelle information nous manque-t-il afin de générer une vidéo 3D ? Pour répondre à cette question, il faut savoir que la vision stéréoscopique binoculaire implique un traitement par le système visuel humain de deux images légèrement différentes. Les différences entre l'image reçue par l'œil droit et celle reçue par l'œil gauche sont des disparités horizontales. Les valeurs de ces disparités dépendent de la profondeur de l'objet, c'est-à-dire de la distance entre l'œil et l'objet regardé. En effet, plus un objet est proche, plus la disparité est grande comme l'illustre la figure 1.1 où la profondeur du crayon bleu est plus petite que celle du crayon gris, contrairement aux disparités.

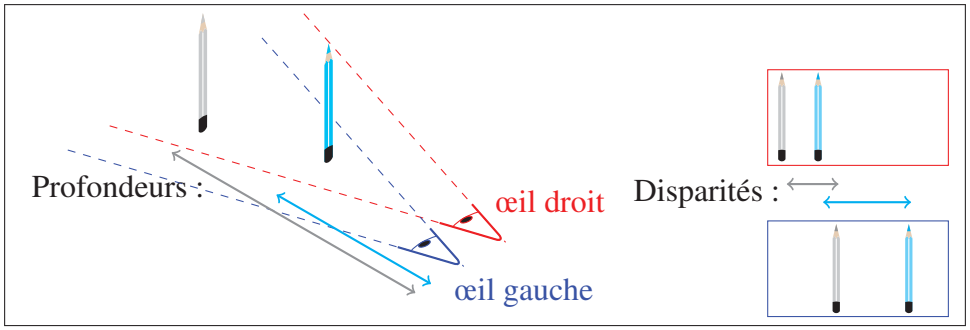


Figure 1.1 Principe de la vision binoculaire

L'affichage 3D sur écran recrée cet effet en envoyant à chaque œil une image différente. Ces deux images peuvent être obtenues par deux caméras identiques, alignées horizontalement et espacées de la même distance que celle qui sépare les deux yeux d'un être humain. Elles peuvent aussi être obtenues par une caméra et des capteurs de profondeur. En effet, comme il y a une relation entre la profondeur et la disparité, il est possible de créer la seconde image de la paire stéréoscopique à partir d'une seule image et l'information de profondeur par l'application d'un algorithme de rendu basé image de profondeur (*depth-image based rendering*, DIBR) (Fehn, 2004).

L'information de profondeur est exprimée sous forme d'une carte de profondeur. Il s'agit d'une image en niveau de gris de même taille que l'image couleur. Sur une carte de profondeur, généralement codée sur 8 bits par pixel, plus la profondeur d'un pixel est grande (respectivement petite) plus la valeur de la carte de profondeur est proche de 0 (respectivement 255). La figure 1.2 montre un exemple.

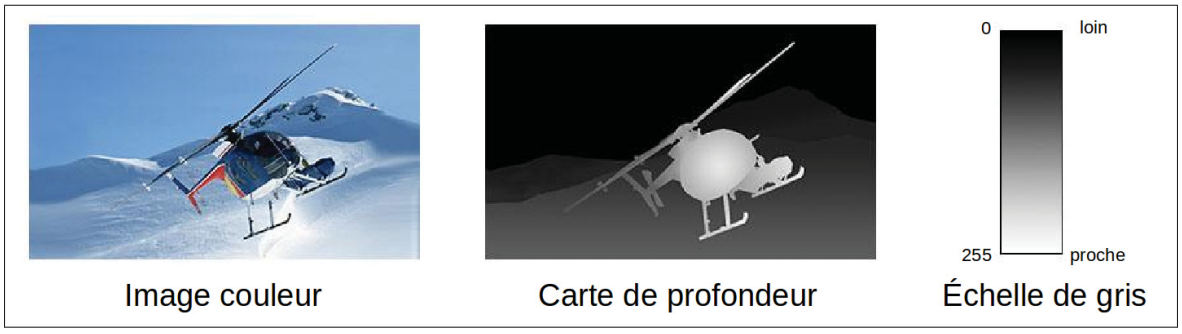


Figure 1.2 Représentation d'une carte de profondeur
Traduite de http://www.i-art3d.com/Eng/About_Depth.htm

Ainsi, la façon la plus répandue pour convertir du contenu 2D en 3D consiste à générer une carte de profondeur pour chaque trame de la vidéo traitée, puis appliquer l'algorithme DIBR. La figure 1.3 résume ce processus.

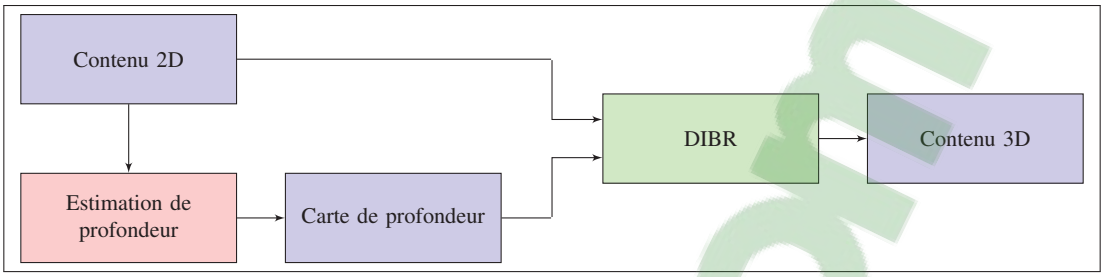


Figure 1.3 Schéma synoptique de la conversion 2D à 3D

1.2 Méthodes de conversion 2D à 3D

Il y a différentes façons d'effectuer une conversion 2D à 3D. En effet, il est possible de l'effectuer manuellement. Cependant, ce travail artistique est lent et par conséquent très coûteux. Afin de réduire les coûts et les délais de cette conversion, des méthodes ont été développées. Elles peuvent être regroupées en deux familles distinctes, à savoir : les méthodes semi-automatiques et les méthodes automatiques. La première famille vise à diminuer l'intervention humaine durant le processus de conversion, alors que la seconde fait exécuter l'ensemble du processus par des machines.

1.2.1 Méthodes semi-automatiques

On retrouve dans la littérature un large éventail d'approches qui permettent la génération semi-automatique de contenu stéréoscopique. Ces approches se distinguent les unes des autres autant par la quantité d'assistance manuelle nécessaire que par les techniques utilisées.
Les notions de trames clés et de propagation de profondeur sont à la base de cette catégorie de méthodes. Il s'agit de sélectionner des trames clés de la vidéo qui recevront un traitement approfondi pour la déduction de la profondeur. Ces trames clés seront espacées, les unes des autres, par d'autres trames. Celles-ci sont non-clés, leurs profondeurs seront déduites par propagation des profondeurs des trames-clés.

Par exemple, dans l'étude de Chen *et al.* (2011) des trames clés sont manuellement sélectionnées, puis chaque trame clé est segmentée par la méthode interactive Watershed (Beu-

cher & Meyer, 1993). Cette segmentation permet de séparer un objet principal de l'arrière-plan. L'intervention humaine permet de marquer par un trait chacune des deux régions, ainsi que d'y attribuer une profondeur. Par la suite, la profondeur est propagée aux trames non-clés en utilisant un modèle actif de contour. Cette approche ne permet d'affecter une profondeur qu'à une seule région. D'autres approches dépassent cette limitation et permettent d'obtenir plusieurs régions à différentes profondeurs. Comme l'approche de Cao *et al.* (2011) qui propose d'effectuer d'abord une sur-segmentation des trames clés par application de l'algorithme Watershed, puis une segmentation multi objets par applications répétés de l'algorithme de Graph-Cut (Boykov *et al.*, 2001). L'assignation des profondeurs est là aussi manuelle. La propagation de la profondeur aux trames non-clés est, quant à elle, élaborée. En effet, en plus du fait qu'elle soit bidirectionnelle, elle introduit la notion de filtre bilatéral décalée qui fait intervenir les informations locales temporelles. L'étude Huang *et al.* (2015) reprend la même méthode utilisée par Cao *et al.* (2011) et y ajoute un procédé de raffinement des cartes de profondeurs propagées. Ce raffinement fait intervenir une modélisation bayésienne basée sur un modèle statistique des scènes naturelles, mais aussi l'application d'une banque de filtres de Gabor à la trame couleur dans l'espace CIELAB et à la profondeur propagée initialement obtenue.

La méthode proposée par Guttmann *et al.* (2009) fait intervenir de la classification. Elle considère comme trames clés la première et dernière trame d'une séquence. Les annotations manuelles consistent en des ensembles de pixels accompagnés d'une profondeur estimée par l'utilisateur. La profondeur des trames clés est obtenue par une optimisation basée sur les moindres carrés pondérés en imposant une contrainte spatiale. Puis, un classifieur de type machine à vecteurs de support (*support vector machine*, SVM) multi classes est entraîné sur les trames clés. Cette classification considère les points clés de type caractéristiques invariantes aux transformations d'échelle (*scale-invariant feature transform*, SIFT) et le niveau de gris de l'image. La classification obtenue est appliquée aux trames non-clés, les prédictions obtenues avec un haut score de confiance sont utilisées comme contrainte dans une étape finale d'optimisation par moindres carrés pondérés. Cette dernière étape prend également en compte des contraintes spatiales, temporelles et les pixels annotés.

Certaines méthodes semi-automatiques utilisent l'information issue d'une estimation de mouvement pour la propagation. Dans ce cadre, les approches de Varekamp & Barenbrug (2007) et Ju *et al.* (2016) traitent de la partie propagation uniquement. La méthode proposée par Varekamp & Barenbrug (2007) consiste dans un premier temps à estimer une profondeur initiale en appliquant un filtre bilatéral, puis à raffiner cette profondeur initiale par une compensation de mouvement. L'estimation de mouvement est effectuée sur des blocs de tailles fixes 16×16 entre la profondeur de la trame précédente et la profondeur initiale. Quant aux travaux de Ju *et al.* (2016), ils utilisent une estimation de mouvement dense. En effet, le flot optique de large déplacement (*large displacement optical flow*, LDOF) est employé (voir section 1.4 pour une description de la méthode). Deux mouvements sont calculés, celui en avant (trame précédente-trame actuelle) et celui en arrière (trame actuelle-trame précédente). Puis la cohérence des deux mouvements est vérifiée. Les pixels où il y a cohérence sont considérés comme régions temporellement cohérentes et leurs profondeurs sont calculées par l'application d'un filtre bilatéral décalé (celui proposé par Cao *et al.* (2011)). L'estimation de la profondeur des régions incohérentes est résolue par un algorithme qui formule le problème sous forme de champs aléatoires de Markov et utilise la saillance structurelle de la trame traitée.

La démarche de Li *et al.* (2012) traite de l'estimation de la profondeur des trames clés et de la propagation. L'étape d'estimation de la profondeur comprend une sur-segmentation par application de l'algorithme des K-moyennes. Puis, une segmentation par application d'un algorithme de max-flow/min-cut sur l'image dont les objets à l'avant et arrière-plan ont été annotés par l'utilisateur. Pour la propagation, l'estimation de mouvement par blocs du standard H.264 est utilisée, et la cohérence avant-arrière est vérifiée (de la même manière que Ju *et al.* (2016)). Ainsi, pour les pixels où celle-ci est établie, la profondeur est obtenue par déplacement de celle de la trame précédente par son mouvement avant. Il en résulte une carte de profondeur contenant des trous qui sont remplis par un filtrage bilatéral. Une dernière étape de raffinement vise à compenser les erreurs dues aux mauvaises estimations de mouvement.

Les travaux Ju & Xiong (2014) et Wang *et al.* (2012) se concentrent sur la sélection des trames clés, à la différence des méthodes précédentes où le choix des trames clés est soit fait manuel-

lement ou par échantillonnage à intervalle régulier. Ju & Xiong (2014) procèdent d'abord à former des clusters de trames consécutives semblables. Ces clusters sont formés par application de l'algorithme des K-moyennes sur l'espace couleur utilisant un histogramme vectoriel de 512 éléments dans l'espace tridimensionnel rouge-vert-bleu (*red-green-blue*, RGB). La première et dernière trame de chaque cluster sont considérées clés (afin d'assurer la possibilité d'une propagation bidirectionnelle), d'autres trames clés y sont au besoin ajoutées selon une analyse de mouvement et jusqu'à occurrence du nombre maximum fixé. Wang *et al.* (2012) détectent dans un premier temps les apparitions de nouvelles scènes en utilisant des histogrammes par blocs, puis la sélection des trames clés parmi les trames d'une même scène se fait en fonction du cumul des occlusions entre trames. Les occlusions sont détectées par application de l'algorithme de stéréo correspondance présenté par Ogale & Aloimonos (2005).

Contrairement aux méthodes précédemment citées, où l'information de profondeur est issue uniquement de l'utilisateur, l'approche Yan *et al.* (2011) intègre, à l'étape de propagation, l'utilisation des occlusions statiques (voir section 1.2.2.1). Les jonctions en T sont détectées de façon semi-automatique. En effet, en plus de spécifier la profondeur de certains pixels des trames clés, il est demandé à l'utilisateur d'indiquer approximativement les régions contenant des jonctions en T. Quant à la méthode proposée par Liao *et al.* (2012), celle-ci vise à minimiser l'intervention humaine puisque l'estimation de profondeur des trames clés est effectuée avant l'intervention de l'utilisateur. Cette estimation est effectuée par des outils qu'utilisent les méthodes automatiques (structure déduite du mouvement (*structure from motion*, SFM), mouvement et perspectives (voir section 1.2.2.1)), ensuite il est demandé à l'utilisateur de confirmer le résultat ou de le compléter s'il y a lieu.

1.2.2 Méthodes automatiques

Les méthodes automatiques reposent sur l'extraction des informations de profondeur présentes dans une vidéo 2D monoscopique. Ces informations sont des indices de profondeurs. Les travaux (Wei, 2005; Zhang *et al.*, 2011) présentent une liste détaillée des indices de profondeurs

utilisés pour la conversion 2D à 3D. La sous-section suivante présente les indices les plus couramment utilisés, suivie d'une revue de littérature des méthodes automatiques.

1.2.2.1 Indices de profondeur

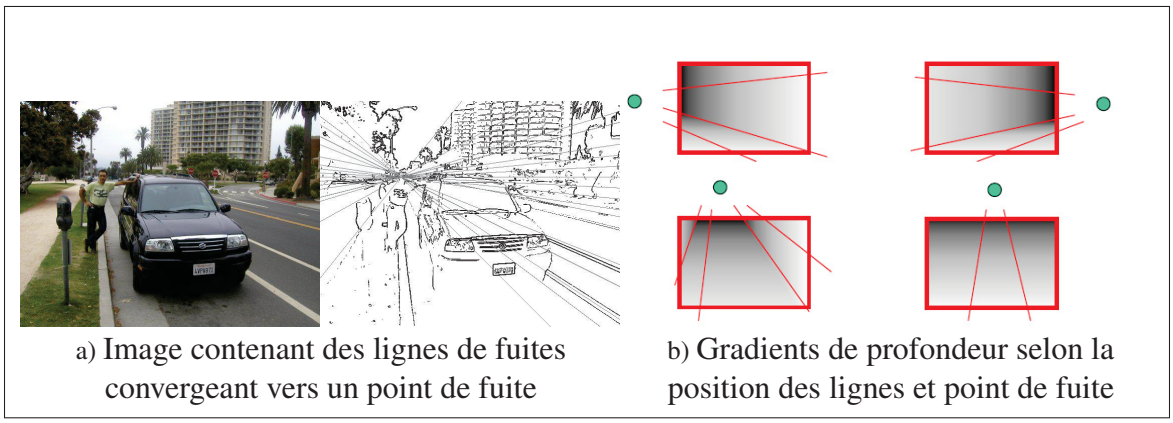
Les indices de profondeur peuvent être classés en deux catégories : les indices statiques extraits à partir d'une seule image et les indices dynamiques qui nécessitent au moins deux images.

1.2.2.1.1 Indices de profondeur statiques

Parmi les indices de profondeur statiques les plus couramment utilisés, il y a :

Couleur et saillance Il s'agit de mettre en avant les parties de l'image qui attirent l'attention de l'observateur. Bien que ça ne soit pas un indice de profondeur à proprement dit, vu qu'un objet peut attirer l'attention même s'il n'est pas en avant-plan (Kim *et al.*, 2010), ce principe est souvent utilisé pour générer un effet de vision stéréoscopique.

Les perspectives linéaires Il s'agit d'exploiter le fait que les lignes parallèles semblent au loin converger (Huang *et al.*, 2009). Le point de convergence est appelé point de fuite. Ainsi, en détectant ces lignes et le point de fuite, un gradient de profondeur peut être affecté à l'image. La figure 1.4 illustre cet indice de profondeur.



a) Image contenant des lignes de fuites convergeant vers un point de fuite

b) Gradients de profondeur selon la position des lignes et point de fuite

Figure 1.4 Perspectives linéaires
Tirées de Battiato *et al.* (2004, p. 6 et 7)

L'accommodation oculaire Cet indice repose sur le fait que plus un objet est loin de la distance focale de la caméra, plus il est flou. L'inconvénient de cet indice est qu'il comporte une ambiguïté dans le sens. En effet, l'éloignement d'une région par rapport au plan focal n'indique pas si elle est en avant ou en arrière du plan focal (Szeliski, 2011). L'accommodation oculaire, souvent appelée focus/defocus, peut être utilisée pour générer la profondeur à partir d'une seule image, comme effectué par l'étude de Tang *et al.* (2013).

Interposition d'objets Une des façons de détecter l'interposition d'objets est la détection des jonctions en T. En effet, lorsque trois régions se superposent, formant une intersection de forme semblable à un T, il est plus probable que la région avec l'angle quasiment plat soit en avant des deux autres (Jia *et al.*, 2012). La figure 1.5 montre un exemple.

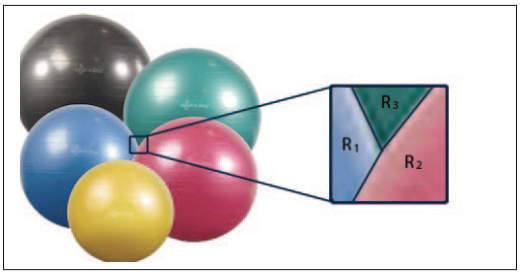


Figure 1.5 Principe des jonctions en T
Tirée de Palou & Salembier (2011, p. 2)

1.2.2.1.2 Indices de profondeur dynamiques

Les principaux indices de profondeur dynamiques sont :

Parallaxe de mouvement Cet indice repose sur l'hypothèse que pour un observateur en mouvement, plus un objet se déplace rapidement, plus il est proche de l'observateur. Ainsi en estimant le mouvement, la profondeur peut être déduite. L'inconvénient est qu'en présence d'objets en mouvement dans la scène observée, l'hypothèse n'est pas toujours vérifiée, ce qui peut mener à des erreurs d'estimation de profondeur.

Une des façons d'utiliser la parallaxe de mouvement est d'appliquer un algorithme de SFM (Jebara *et al.*, 1999). Il s'agit d'utiliser plusieurs images d'une séquence vidéo, où

il est supposé que la caméra est en mouvement et que la scène est fixe. Une reconstruction 3D éparse est obtenue par minimisation d'erreur de rétroprojection de points clés en correspondances sur les images traitées. La figure 1.6 illustre le principe.

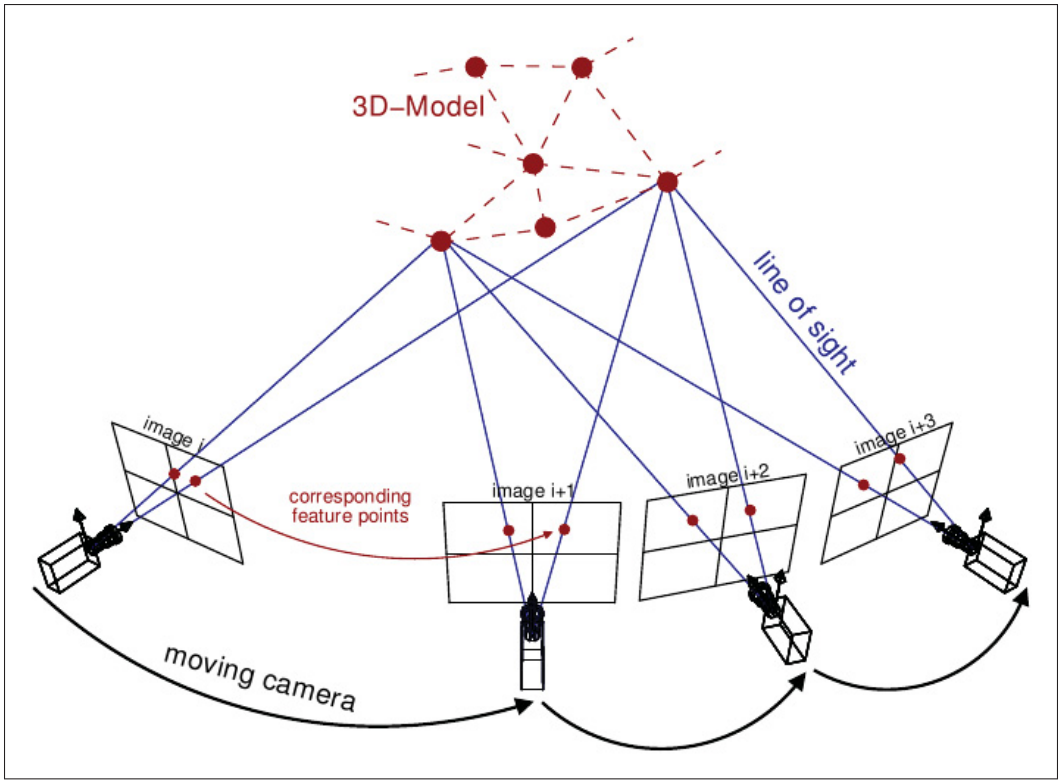


Figure 1.6 Principe des approches SFM
Tirée de http://www.theia-sfm.org/sfm.html

Occlusions dynamiques Les occlusions dynamiques réfèrent au fait qu'un objet en mouvement cache une partie de ce qui se trouve en arrière de lui.

1.2.2.2 Revue de littérature des méthodes automatiques

Parmi les approches qui utilisent uniquement des indices statiques, on peut citer celle de Chen & Huang (2016) qui traite des images seules. La première étape de cette méthode consiste à déterminer un gradient de profondeur selon la localisation des points de fuite ainsi qu'effectuer une segmentation basée sur l'arbre recouvrant de poids minimal. Puis le gradient et la

segmentation obtenus sont combinés pour générer une carte de profondeur initiale. La détection d'objets saillants permet ensuite de raffiner ce résultat. Zhang *et al.* (2015) proposent une méthode temps réel basée uniquement sur la couleur. La saillance de chaque couleur est calculée, puis une profondeur est attribuée en fonction de la valeur de celle-ci. Plus une couleur est saillante plus elle sera considérée proche. Afin d'assurer la cohérence en temps des cartes de profondeurs générées, cette méthode détecte les nouvelles scènes à l'aide d'un histogramme de couleur, et le calcul de la saillance des couleurs ne se fait qu'une seule fois par scène. L'approche suivie par Yin *et al.* (2015) intègre également la détection de nouvelles scènes pour assurer la cohérence en temps. Suivant s'il y a détection du ciel et/ou de point de fuites, chaque scène est classifiée dans un des trois types suivant : «paysage», «perspective linéaire» ou «normal». Selon le résultat de classement, une carte de profondeur globale est déterminée, puis celle-ci est raffinée en utilisant la couleur. Ce raffinement consiste à combiner linéairement l'ordre global calculé et les composantes Y et Cr de l'espace de couleur YCrCb.

Jung *et al.* (2015) utilise uniquement un indice de profondeur dynamique : la parallaxe de mouvement. Le mouvement est calculé par une méthode de flot optique (voir section 1.4). Selon le nombre de vecteurs de mouvement nuls du flot optique, le mouvement est considéré local ou global. Si le mouvement est global alors le principe de la parallaxe de mouvement est appliqué uniformément à toute l'image. Sinon, dans le cas de mouvement local, un gradient de profondeur est combiné aux parallaxes de mouvement afin d'attribuer des profondeurs aux pixels dont le mouvement est nul.

Les études (Knorr *et al.*, 2007; Zhang *et al.*, 2008; Liu *et al.*, 2015b) appliquent des approchent de structure déduite du mouvement (*structure from motion*, SFM) pour effectuer la conversion. Ils proposent différentes techniques pour passer d'une carte de profondeur éparse obtenue par SFM à une carte de profondeur dense. En effet, pour effectuer ce passage, (Zhang *et al.*, 2008) et (Liu *et al.*, 2015b) utilisent des segmentations couleur alors que (Knorr *et al.*, 2007) modélise le passage en champ aléatoire de Markov. Knorr *et al.* (2007) tentent de contourner les limitations des algorithmes SFM qui sont adaptés aux vidéos contenant des scènes fixes filmées

par des caméras en mouvement, en intégrant à l'algorithme SFM les informations de détection et de suivi d'objets indépendamment mobiles dans la scène.

Plusieurs méthodes combinent des indices statiques et dynamiques. Par exemple, l'étude de Cheng & Sung (2013) propose une combinaison d'indices de profondeur pour générer une carte de profondeur relative par blocs de taille 16×16. Les indices statiques utilisés sont la clarté, le contraste et la couleur. La parallaxe de mouvement est utilisée comme indice de profondeur dynamique. L'approche Lai *et al.* (2013) consiste, dans un premier temps, à séparer l'arrière-plan statique des objets mobiles par seuillage de la différence absolue des valeurs de pixel entre deux trames consécutives. Puis, les lignes de fuite de la trame traitée sont détectées par application d'un algorithme de Hough simplifié. Selon la position et le nombre de lignes de fuites détectées, un gradient de profondeur est attribué à l'image. La profondeur est obtenue par pondération du gradient de profondeur et de la luminance de la trame. Cette pondération varie selon que le pixel soit dans la zone des objets qui se déplacent ou dans la zone d'arrière-plan. Ainsi, plus de poids est donné au gradient de profondeur dans la zone arrière-plan, et plus de poids est donné à la texture pour les pixels des objets en mouvement.

Une sous-famille des méthodes automatiques est la famille des méthodes basées sur l'apprentissage. Dans cette catégorie, l'étude (Konrad *et al.*, 2013) propose deux méthodes. Une méthode d'apprentissage locale, qui permet d'affecter une profondeur à chaque pixel selon sa couleur, sa position et son mouvement. Ainsi qu'une méthode d'apprentissage globale, qui se base sur la recherche de K plus proches voisins d'une image selon son histogramme de couleur dans une large base de données contenant des images couleur et leurs cartes de profondeurs. La profondeur de l'image est obtenue par la médiane et filtrage des cartes de profondeurs des K plus proches images. (Karsch *et al.*, 2015) suit une démarche semblable à cette seconde méthode et des améliorations y sont proposées. En effet, SIFT flow (Liu *et al.*, 2015a) y est utilisé pour aligner les K plus proches images à l'image traitée, et une optimisation plus élaborée qu'un calcul de médian est effectuée pour combiner les cartes de profondeurs alignées. Une extension de cette méthode pour le traitement de vidéo est également explorée, elle comprend la détection d'objets en mouvement et l'utilisation du flot optique. L'approche proposée par (Her-

rera *et al.*, 2016) introduit une étape hors ligne qui classifie la base de données d'apprentissage en clusters. Cette classification utilise une combinaison de quatre descripteurs : histogrammes des gradients orientés (*histograms of oriented gradients*, HOG), patrons binaires locaux (*local binary patterns*, LBP), caractéristiques robustes accélérées (*speeded-up robust features*, SURF) et GIST (Oliva & Torralba, 2001). Elle permet d'accélérer la recherche des images semblables à l'image traitée. La carte de profondeur est estimée par fusion des profondeurs sélectionnées puis raffinée par l'application d'un filtre de lissage adaptatif qui fait intervenir une segmentation couleur. Sun *et al.* (2009) utilisent l'apprentissage pour classifier la scène selon deux catégories : scènes orientées objet, et scènes non orientées objet. Puis une profondeur initiale est générée, par analyse de contexte dans le cas d'une scène non orientée objet et par extraction d'objets dans le cas d'une scène orientée objet. Puis cette profondeur initiale est fusionnée à la profondeur obtenue grâce à l'indice de focus. Xie *et al.* (2016) proposent, quand à eux, une méthode basée sur l'apprentissage profond. Cette méthode vise à générer à partir d'une image considérée comme point de vue gauche, l'image de point de vue droit, sans passer par l'estimation d'une carte de profondeur précise, en exploitant des réseaux de neurones convolutionel (*convolutional neural networks*, CNN) entraînés sur des paires d'images stéréoscopiques.

Comme le présent travail traite des occlusions, une revue de littérature détaillée sur ce sujet est présentée à la section 1.3.

1.3 Les occlusions

Cette section comporte deux parties. La première traite des recherches utilisant des occlusions statiques et dynamiques pour la conversion 2D à 3D. La seconde aborde l'utilisation des occlusions dans d'autres cadres que la conversion 2D à 3D.

1.3.1 Exploitation des occlusions pour la conversion 2D à 3D automatique

1.3.1.1 Occlusions statiques

Plusieurs recherches ont exploré les occlusions statiques pour la génération de cartes de profondeur. Ces occlusions sont en général exprimées sous forme de jonctions en T (Jia *et al.*, 2012; Ming *et al.*, 2016; Liu *et al.*, 2013; Dimiccoli & Salembier, 2009; Rezaeirowshan *et al.*, 2015; Palou & Salembier, 2013). Les jonctions en T permettent de déduire un ordre local entre leurs trois régions. Les recherches qui sont menées dans ce domaine se différencient les unes des autres par la manière dont les jonctions en T sont détectées, par l'approche suivie pour obtenir une segmentation, ainsi que par la façon dont le passage entre ordre local et ordre global est déduit.

Certaines de ces recherches se basent sur l'apprentissage. Dans ce contexte, on peut citer Jia *et al.* (2012) qui considèrent des caractéristiques des jonctions en T ainsi que des caractéristiques de contours pour entraîner des classifieurs SVM. Cette approche utilise un champ aléatoire de Markov pour effectuer le passage de l'ordre local à l'ordre global. Ming *et al.* (2016) utilisent l'apprentissage pour classer les bords dans une des deux catégories : bords triviaux ou bords d'occlusion. Les bords de départ sont déterminés par une sur-segmentation initiale. Des caractéristiques de dimension 92 incluant les informations de couleur, de texture et de contour caractérisent chaque bord. Le résultat de l'apprentissage permet d'identifier les bords d'occlusion. Une fois les bords d'occlusion sélectionnés, un triple descripteur comprenant deux jonctions en T et un segment de courbure les reliant permet la déduction d'un ordre local. L'utilisation du triple descripteur permet d'éliminer certaines relations d'ordre local erronées, ce que ne permettait pas l'approche de Jia *et al.* (2012). Les ordres locaux sont donc plus fiables. Il en résulte une déduction d'ordre global plus simple.

La méthode proposée par Liu *et al.* (2013) s'intéresse à la conversion automatique de dessins animés réalisés à la main. Les jonctions en T sont les seuls indices de profondeur considérés vu leur fiabilité dans le contexte des dessins manuels. Une segmentation est effectuée, celle-

ci tire avantage du fait que dans un dessin manuel les contours sont plus foncés que le reste de l'objet. La déduction de l'ordre global est formulée sous forme de Graph-Cut. Bien que la méthode considère les occlusions statiques au niveau de chaque image, toutes les images d'une séquence sont conjointement traitées et considérées lors de l'étape de la déduction de l'ordre. Il en résulte une cohérence temporelle des cartes de profondeurs générées.

Parmi les recherches qui traitent des contenus capturés par des caméras, et sans avoir recours à l'apprentissage, celle de Dimiccoli & Salembier (2009) propose une méthode de calcul robuste des jonctions en T. Une segmentation préservant ces jonctions est effectuée. Les jonctions en T permettent de déduire un ordre local entre régions adjacentes, cet ordre peut être représenté sous forme d'un graphe dirigé. L'ordre global est déduit en deux étapes. La première étape consiste à transformer le graphe en un graphe dirigé acyclique par suppression des arcs les moins fiables. La seconde étape est une réduction transitive de ce graphe. Cette réduction transitive permet d'obtenir un diagramme de Hasse qui représente la carte de profondeur recherchée. Contrairement à Dimiccoli & Salembier (2009), l'approche Rezaeirowshan *et al.* (2015) traite des images déjà segmentées. Elle combine un indice reflétant la convexité des régions segmentées et les jonctions en T pour établir un ordre entre les objets délimités par la segmentation. Un ordre partiel est déduit entre les objets adjacents, puis un ordre global est calculé au moyen d'une méthode d'agrégation de rangs (Dekel (Basha) *et al.*, 2014).

La méthode suivie par Palou & Salembier (2013) se base sur les jonctions en T et la convexité pour établir l'ordre entre régions. La segmentation est obtenue par la construction d'un arbre de partition binaire (*binary partition tree*, BPT) puis l'élagage de celui-ci. L'ordre global est déduit en effectuant un ordonnancement topologique partiel d'un graphe d'ordre de profondeur dont les conflits d'ordre ont été résolus. Ces étapes (construction d'un BPT, élagage, et déduction d'ordre global) sont similaires aux étapes suivies par les études Palou (2013); Salembier & Palou (2014) qui traient des occlusions dynamiques. (voir section 1.5).

1.3.1.2 Occlusions dynamiques

Schodl & Essa (2001) traitent le cas d'une scène statique ne contenant qu'un seul objet qui se déplace. Cet objet est à la fois caché par des parties de la scène et cache lui-même d'autres parties de celle-ci. L'approche effectue une soustraction d'arrière-plan, puis à chaque trame, les régions n'appartenant pas à l'arrière-plan sont considérées comme étant des parties de l'objet en mouvement qui cachent des régions de l'arrière-plan. L'analyse à travers le temps des partie de l'arrière-plan caché permet d'ordonner les différentes régions de l'arrière-plan.

Kowdle *et al.* (2013) combinent l'utilisation d'occlusions dynamiques et les occlusions statiques pour traiter le cas où plusieurs objets se déplacent dans un environnement capturé par une caméra fixe. Cette approche tire avantage de la configuration en caméra fixe. En effet, cette configuration permet aisément l'extraction de l'arrière-plan (par calcul du médian des trames de la scène), ce qui facilite les déductions d'ordre entre les objets en mouvement et les différentes parties de l'arrière-plan.

D'autres méthodes s'inscrivent dans un cadre moins contraignant, où il n'y a pas de restrictions sur le mouvement possible de la caméra, ni sur le nombre d'objets contenus dans une scène. Par exemple, Feng *et al.* (2011) proposent une méthode de conversion basée objets. En effet, ils utilisent le flot optique pour obtenir une segmentation et un ordre des objets contenus dans une image. Le flot optique utilisé est calculé selon l'approche de Gautama & Van Hulle (2002). Les pixels avec des flot optiques semblables sont regroupés dans une même région. L'application du flot optique à la région leur permet de détecter les zones cachées dans la trame traitée par rapport à la trame qui la suit et d'en déduire l'ordre des objets. La segmentation obtenue par flot optique est ensuite raffinée en considérant l'intensité des pixels de l'image. La fusion de cette segmentation et de l'ordre déduit à la première étape permet d'obtenir une carte de profondeur finale. La méthode proposée par Palou (2013) s'inscrit dans ce même cadre, le flot optique est utilisé pour détecter les zones cachées et déduire l'ordre. Contrairement à Feng *et al.* (2011), où la segmentation se fait en deux étapes, une première qui ne considère que le flot optique et une seconde qui ne prend en compte que l'intensité des pixels, la méthode proposée par

Salembier & Palou (2014) permet de considérer conjointement la couleur et le mouvement pour l'obtention de régions segmentées pouvant être ordonnées. Cette méthode est détaillée à la section 1.5.

1.3.2 Utilisation des occlusions à d'autres fins

L'étude Fu *et al.* (2016) vise à détecter les bordures d'occlusions par une exploration de contexte. L'approche utilise des CNN et y intègre deux caractéristiques d'occlusion. La première caractéristique est la discontinuité du flot optique qui est exprimée par le module du gradient du flot optique. La seconde caractéristique, quant à elle, exprime l'incohérence entre flot optique avant et flot optique arrière. Le principe de cette incohérence est illustré à la figure 1.7. Fu *et al.* (2016) expriment l'incohérence en combinant la distance euclidienne entre le point de départ et le point de retour (distance entre point rouge et point jaune sur la figure 1.7) ainsi que la différence d'angle entre le flot avant et le flot arrière (θ sur la figure 1.7).
On peut remarquer que l'utilisation du flot optique est récurrente pour la conversion 2D à 3D et/ou pour détecter les occlusions. La section suivante présente cet outil et détaille trois algorithmes de l'état de l'art permettant son estimation.

1.4 Flot optique

Le flot optique est une méthode d'estimation de mouvement dense entre deux images. L'état de l'art offre une multitude d'algorithmes permettant le calcul du flot optique. Les sous-sections suivantes présentent une méthode classique de flot optique, le flot optique de large déplacement (*large displacement optical flow*, LDOF) développé par Brox & Malik (2011) et utilisé par la méthode de Salembier & Palou (2014), ainsi qu'un algorithme plus récent, l'EpicFlow (Revaud *et al.*, 2015).
Soient, pour les sous-sections suivantes :

- $I_t(\mathbf{x})$ et $I_{t+1}(\mathbf{x})$ deux images consécutives, où $\mathbf{x} = (x, y)$ exprime la localisation des pixels de l'image dans son domaine rectangulaire Ω, $\Omega \subset \mathbb{N}^2$, et t est le temps.

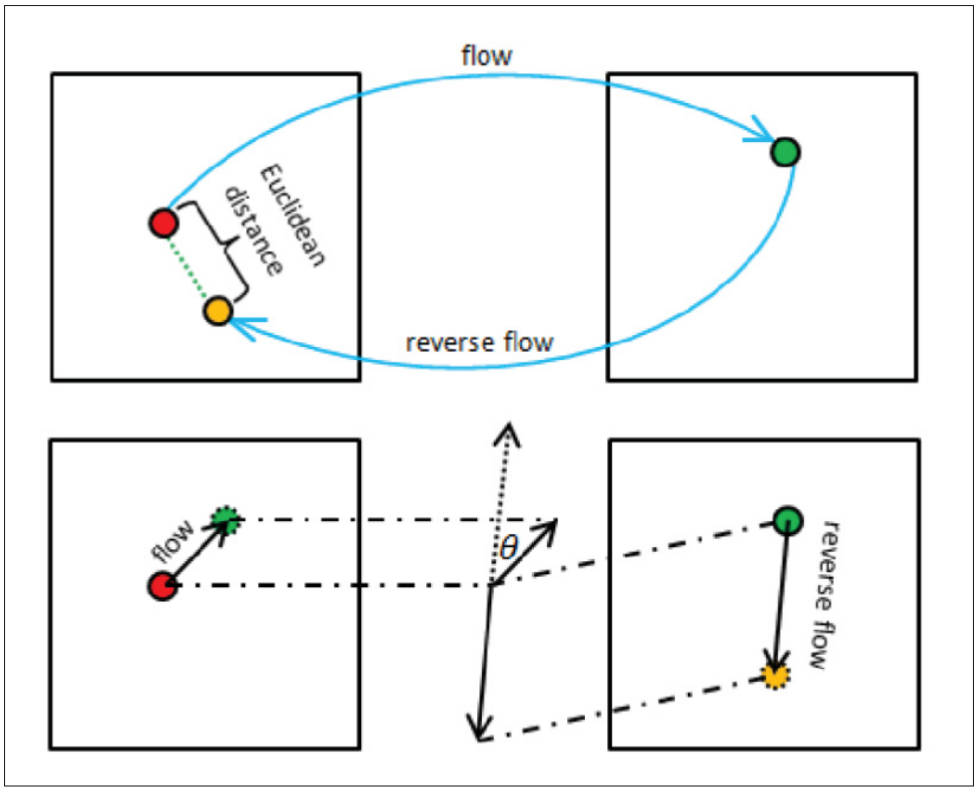


Figure 1.7 Illustration de l'incohérence avant-arrière du flot optique
Tirée de Fu *et al.* (2016, p. 5)

- $\mathbf{w}(\mathbf{x}) = (u, v)$ un champs vectoriel, représentant le flot optique de l'image I_t à I_{t+1}. Avec $u \in \mathbb{R}$ et $v \in \mathbb{R}$, représentant respectivement le déplacement horizontal et vertical du pixel $\mathbf{x}$.

1.4.1 Flot optique classique

Afin de calculer le mouvement de chaque pixel d'une image, les algorithmes d'estimation du flot optique effectuent la minimisation de la différence de luminance ou de couleur entre les pixels que le flot optique fait correspondre. Cette contrainte peut être exprimée par la minimisation de l'expression suivante (Szeliski, 2011) :

$$\min\ E(\mathbf{w}) \quad \text{avec} \quad E(\mathbf{w}) = \sum_{\mathbf{x}\in\Omega} \left[I_{t+1}(\mathbf{x}+\mathbf{w}) - I_t(\mathbf{x})\right]^2 \ , \mathbf{w} \in \mathbb{R}^2 \tag{1.1}$$

Ce système étant sous-contraint, des contraintes lui sont ajoutées, ce qui rend sa résolution unique et possible. La contrainte généralement ajoutée est une contrainte de lissage ou de constance de gradient, qui suppose qu'il n'y a pas de changement brut de mouvement. Par exemple, pour la méthode (Horn & Schunck, 1981), le calcul du flot optique s'effectue par minimisation de la fonctionnelle ci-dessous :

$$E(\mathbf{w}) = \int_{\mathbf{x}\in\Omega} \underbrace{\left(\nabla I_t + I_t'\right)^2}_{E_{Couleur}} + \underbrace{\alpha^2(\|\nabla u\|^2 + \|\nabla v\|^2)}_{E_{Lissage}} \mathrm{d}\mathbf{x} \tag{1.2}$$

Avec I_t' la dérivée de l'image I_t par rapport au temps, et l'opérateur ∇ désigne le gradient. $E_{Couleur}$ s'exprime sous cette forme, car la constance de luminance suppose :

$$\frac{dI}{dt} = 0 \tag{1.3}$$

En développant, l'expression devient :

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{1.4}$$

Comme $u = \frac{dx}{dt}$ et $v = \frac{dy}{dt}$, alors :

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + I_t' = 0 \tag{1.5}$$

Ce qui peut s'écrire

$$\begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}^T \cdot \begin{pmatrix} u \\ v \end{pmatrix} + I_t' = 0 \quad \Rightarrow \quad \nabla I + I' = 0. \tag{1.6}$$

1.4.2 Flot optique de large déplacement

Le flot optique de large déplacement (Brox & Malik, 2011) est une approche qui, en plus de considérer la couleur et les contraintes de lissage et de constance de gradient, intègre l'utili-

sation de points de correspondances. Les points de correspondances considérés sont ceux du descripteur HOG (Dalal & Triggs, 2005).

Le flot optique est obtenu par minimisation de la fonctionnelle ci-dessous :

$$E(\mathbf{w}) = \underbrace{\int_\Omega \Psi\left(|I_{t+1}(\mathbf{x}+\mathbf{w}(\mathbf{x})) - I_t(\mathbf{x})|^2\right) \mathrm{d}x}_{E_{Couleur}} + \gamma \underbrace{\int_\Omega \Psi\left(|\nabla I_{t+1}(\mathbf{x}+\mathbf{w}(\mathbf{x})) - \nabla I_t(\mathbf{x})|^2\right) \mathrm{d}x}_{E_{Gradient}} + \alpha \underbrace{\int_\Omega \Psi\left(|\nabla u(\mathbf{x})|^2 + |\nabla v(\mathbf{x})|^2\right) \mathrm{d}x}_{E_{Lissage}} + \beta \underbrace{\int_\Omega \delta(\mathbf{x})\rho(\mathbf{x})\Psi\left(|\mathbf{w}(\mathbf{x}) - \mathbf{w}_{desc}(\mathbf{x}))|^2\right) \mathrm{d}x}_{E_{Correspondances}} \tag{1.7}$$

Ψ étant la fonction robuste : $\Psi(s^2) = \sqrt{s^2+\varepsilon^2}$ avec $\varepsilon = 0.001$, α, β et γ sont des facteurs de pondération.

Le terme $E_{Correspondances}$ permet de prendre en compte les points mis en correspondance par le descripteur HOG. La fonction $\delta(\mathbf{x})$ indique la mise en correspondance ou non du point $\mathbf{x}$, $\rho(\mathbf{x})$ la confiance de cette mise en correspondance et $\mathbf{w}_{desc}(\mathbf{x})$ le déplacement du point $\mathbf{x}$ pour arriver à son point correspondant en I_{t+1}.

L'intégration des descripteurs HOG permet d'obtenir des résultats beaucoup plus précis que ne le fait une approche classique. Mais, la régularisation (introduite par le terme $E_{Lissage}$, pondérée par le facteur α) produit un effet de lissage sur le flot optique obtenu et ne permet pas de conserver les contours.

1.4.3 EpicFlow

L'EpicFlow (Revaud *et al.*, 2015) est une méthode de flot optique conçue dans le but de préserver les contours. La figure 1.8 illustre son principe et ses étapes.

La méthode repose sur l'hypothèse que la plus grande partie des contours de mouvements sont aussi des contours de couleurs. De ce fait, pour calculer le mouvement d'une image I_t à une image I_{t+1}, les contours de couleurs de l'image I_t sont calculés en utilisant le détecteur de contours structuré (*structured edge detector*, SED) (Dollár & Zitnick, 2015). Cette méthode renvoie une image $C_t(\mathbf{x})$, qui pour chaque pixel $\mathbf{x}$ contient une valeur entre 0 et 1, représentant

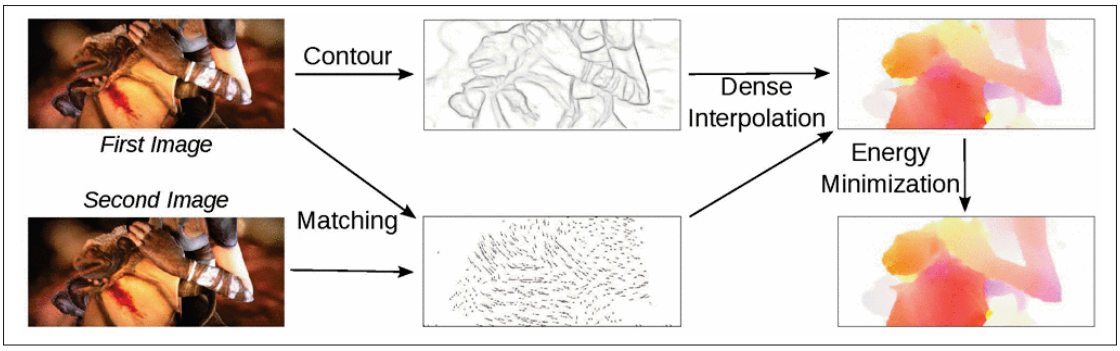


Figure 1.8 Étapes de l'EpicFlow
Tirée de Revaud *et al.* (2015, p. 2)

la force du contour à ce pixel. La valeur 0 exprime qu'un pixel n'est pas un contour, plus cette valeur est élevée, plus il est probable que le pixel soit un contour.

Des correspondances $(\mathbf{x}_m, \mathbf{x}_{m'})$ avec $\mathbf{x}_m \in I_t$ et $\mathbf{x}_{m'} \in I_{t+1}$ de points entre l'image I_t et I_{t+1} sont calculés au moyen de l'algorithme deepMatching (Weinzaepfel *et al.*, 2013). Cet algorithme a pour avantage de faire correspondre plus de points clés que d'autres algorithmes de la littérature (SIFT (Lowe, 2004), SURF (Bay *et al.*, 2006), etc.).
Le mouvement est donc connu aux points mis en correspondance. Une interpolation est effectuée pour attribuer un correspondant $F(\mathbf{x})$ à chaque pixel $\mathbf{x}$ de l'image I_t. En plus des points de correspondance trouvés, cette interpolation utilise l'information de contour. En effet, pour chaque pixel, les K plus proches points de correspondance sont considérés pour l'interpolation selon une distance géodésique $D_G(\mathbf{x}, \mathbf{x}_m)$ à partir de C_t.
Comme le calcul de la distance géodésique est un processus itératif coûteux en nombre d'opérations, une approximation est introduite. Cette approximation s'effectue par le calcul de cellules de Voronoï. En effet, un partitionnement L est caculé, assignant à chaque pixel $\mathbf{x} \in I_t$ sont plus proche point de correpodance $\mathbf{x}_m$. Puis un graphe G est construit, chaque nœud de ce graphe représente une cellule de Voronoï, chaque paire de nœuds dont les régions respectives sont adjacentes est reliée par un arc. Le poids de cet arc est égal à la distance géodésique entre les centres des deux régions reliées. La distance entre deux points de correspondances $(\mathbf{x}_a, \mathbf{x}_b)$ dont les régions ne sont pas adjacentes est calculée à l'aide de l'algorithme Dijkstra ($\tilde{D}_G(\mathbf{x}_a, \mathbf{x}_b) = D_G^{\mathscr{G}}(\mathbf{x}_a, \mathbf{x}_b)$). La distance entre un pixel $\mathbf{x}$ et un point de correspondance $\mathbf{x}_m$ est

approximée par la somme de la distance entre le pixel $\mathbf{x}$ et son plus proche correspondant $L(\mathbf{x})$ ainsi que la distance entre ce plus proche voisin $L(\mathbf{x})$ et le point de correspondance $\mathbf{x}_m$:

$$\tilde{D}_G(\mathbf{x},\mathbf{x}_m) = D_G(\mathbf{x},L(\mathbf{x})) + D_G^{\mathscr{G}}(L(\mathbf{x}),\mathbf{x}_m) \tag{1.8}$$

Deux options d'interpolation sont proposées par Revaud *et al.* (2015), la première utilise une estimation Nadaraya-Watson (*Nadaraya-Watson*, NW)(Wasserman, 2004), la seconde une estimation affine localement-pondérée (*locally-weighted affine*, LA) (Hartley & Zisserman, 2003).

Pour l'interpolation NW, le point de correspondance attribué à chaque pixel est une somme pondérée de ses correspondances voisines, comme exprimé par l'équation 1.9.

$$F_{NW}(\mathbf{x}) = \frac{\sum\limits_{(\mathbf{x}_m,\mathbf{x}'_m)\in\mathscr{N}_K(L(\mathbf{x}))} k_D(\mathbf{x},\mathbf{x}_m)\mathbf{x}'_m}{\sum\limits_{(\mathbf{x}_m,\mathbf{x}'_m)\in\mathscr{N}_K(L(\mathbf{x}))} k_D(\mathbf{x},\mathbf{x}_m)} \tag{1.9}$$

$\mathscr{N}_K(L(\mathbf{x}))$ étant des paires $(\mathbf{x}_m,\mathbf{x}_{m'})$ de correspondances, où les $\mathbf{x}_m$ sont les K plus proches points mis en correspondance de $L(\mathbf{x})$.
Et avec $k_D(\mathbf{x},\mathbf{x}_m) = exp^{-aD_G(\mathbf{x},\mathbf{x}_m)}$ un noyau Gaussien de la distance D_G de paramètre a.
En remplaçant D_G par son expression de l'équation 1.8, k_D devient :

$$k_D(\mathbf{x},\mathbf{x}_m) \simeq k_{\tilde{D}_G}(\mathbf{x},\mathbf{x}_m) = k_{D_G}(\mathbf{x},L(\mathbf{x})) \times k_{D_G^{\mathscr{G}}}(L(\mathbf{x}),\mathbf{x}_m) \tag{1.10}$$

Puis, en remplaçant l'équation 1.10 dans l'équation 1.9, F_{NW} devient :

$$\begin{aligned} F_{NW}(\mathbf{x}) &= \frac{\sum\limits_{(\mathbf{x}_m,\mathbf{x}'_m)\in\mathscr{N}_K(L(\mathbf{x}))} k_D(\mathbf{x},\mathbf{x}_m)\mathbf{x}'_m}{\sum\limits_{(\mathbf{x}_m,\mathbf{x}'_m)\in\mathscr{N}_K(L(\mathbf{x}))} k_D(\mathbf{x},\mathbf{x}_m)} \\ &= \frac{k_{D_G}(\mathbf{x},L(\mathbf{x}))\sum\limits_{(\mathbf{x}_m,\mathbf{x}'_m)\in\mathscr{N}_K(L(\mathbf{x}))} k_{D_G^{\mathscr{G}}}(L(\mathbf{x}),\mathbf{x}_m)\mathbf{x}'_m}{k_{D_G}(\mathbf{x},L(\mathbf{x}))\sum\limits_{(\mathbf{x}_m,\mathbf{x}'_m)\in\mathscr{N}_K(L(\mathbf{x}))} k_{D_G^{\mathscr{G}}}(L(\mathbf{x}),\mathbf{x}_m)} = F_{NW}(L(\mathbf{x}) \end{aligned} \tag{1.11}$$

Quant à l'interpolation LA, elle utilise une estimation affine localement pondérée :

$$F_{LA}(\mathbf{x}) = A_{\mathbf{x}}\mathbf{x} + t_{\mathbf{x}}^{\top} \tag{1.12}$$

Avec $A_{\mathbf{x}}$ et $t_{\mathbf{x}}^{\top}$ calculés en résolvant, par la méthode des moindres carrés, le système comportant pour chaque paire $(\mathbf{x}_m, \mathbf{x}_{m'}) \in \mathcal{N}_K(L(\mathbf{x}))$ l'équation :

$$k_D(\mathbf{x}, \mathbf{x}_m)(A_{\mathbf{x}}\mathbf{x} - m + t_{\mathbf{x}}^{\top} - \mathbf{x}'_m) = 0 \tag{1.13}$$

En tenant compte, de l'équation 1.10, F_{LA} devient :

$$F_{LA}(\mathbf{x}) = A_{L(\mathbf{x})}\mathbf{x} + t_{L(\mathbf{x})}^{\top} \tag{1.14}$$

Ainsi les paramètres A et t ne sont calculés que pour les points de correspondances $\mathbf{x}_m$.

L'interpolation est suivie par une étape de raffinement par minimisation d'énergie.

L'EpicFlow donne de très bons résultats, préserve les contours, et son implémentation est optimisée pour que son temps de calcul soit moindre. Seulement la cohérence avant-arrière n'est pas assurée. Pour rendre ce flot optique cohérent avant-arrière, la présente étude propose d'y apporter des modifications. Ces modifications sont détaillées au chapitre 2, section 2.2.

1.5 Méthode de Salembier et Palou pour l'estimation de profondeur basée sur les occlusions dynamiques

La méthode proposée par Salembier & Palou (2014), utilise le flot optique LDOF pour estimer les occlusions et l'ordre des différentes régions de l'image. Les régions de l'image sont obtenues par construction d'un arbre BPT, puis élagage de celui-ci. La figure 1.9 illustre les différentes étapes de la méthode. Ces étapes sont détaillées aux sous-sections suivantes.

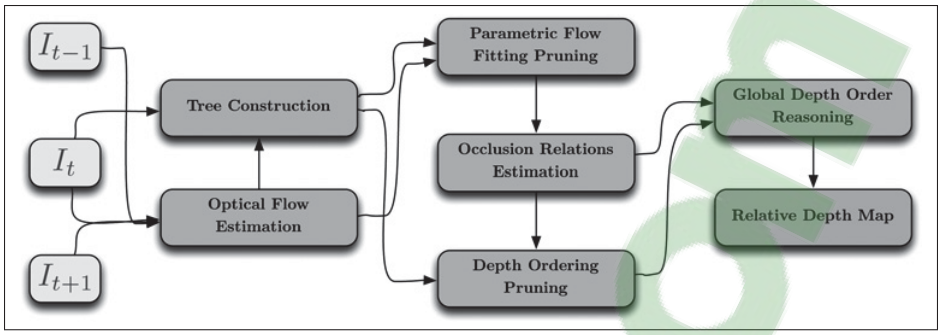


Figure 1.9 Étapes de la méthode (Salembier & Palou, 2014)
Tirée de Salembier & Palou (2014, p. 2)

1.5.1 Construction de l'arbre de partition binaire

L'arbre BPT a été introduit par Salembier & Garrido (2000). Il s'agit d'une structure hiérarchique de représentation de données sous forme d'arbre binaire. Les feuilles de l'arbre représentent des éléments des données originales, et les autres nœuds représentent la fusion de ses deux nœuds fils (Alonso-González, 2014).

Salembier & Palou (2014) utilise un BPT pour représenter la trame traitée. Chaque feuille représente un pixel de cette trame, et les autres nœuds représentent la fusion de deux régions voisines. Ainsi, par construction, le nœud racine représente la trame au complet.
La construction de cet arbre se fait selon une approche ascendante, en fusionnant de façon itérative les deux régions voisines les plus similaires. Pour connaître à chaque itération les régions voisines, un graphe de régions d'adjacence (*region adjacency graph*, RAG) est construit et mis à jour à chaque itération de la construction du BPT. La figure 1.10 illustre ce processus.
Un RAG est un graphe non orienté pondéré, où les nœuds représentent les régions de l'image. Les arcs du RAG relient les régions voisines, le poids de chaque arc représente la mesure de similarité des deux régions qu'il relie. Le calcul de cette mesure de similarité est détaillé à la sous-section 1.5.1.1.

À chaque itération de la construction de l'arbre, le RAG est mis à jour, en remplaçant les deux nœuds des régions venant d'être fusionnées par un nœud représentant la fusion des deux

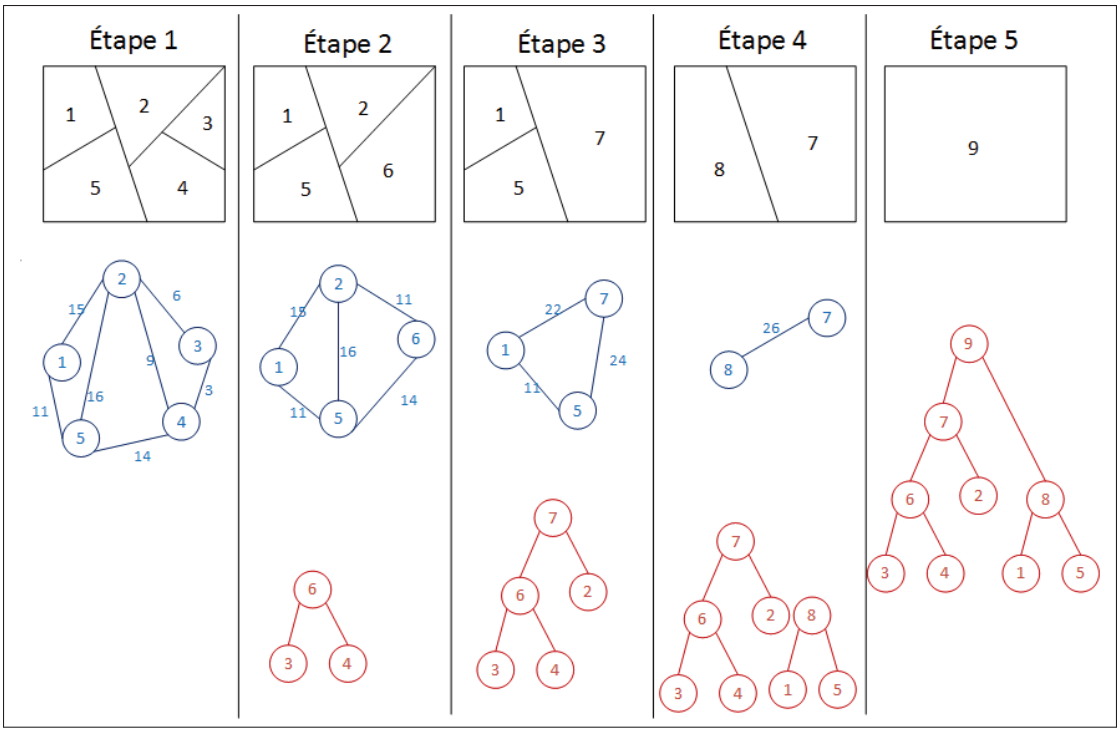


Figure 1.10 Illustration du processus de formation d'un BPT. À chaque étape le graphe RAG est représenté en bleu et l'arbre BPT en rouge

nœuds. Les arcs devant relier ce nouveau nœud à ses nœuds voisins sont ajoutés et leurs poids calculés. La figure 1.11 résume les étapes de la construction du BPT.

Une fois l'arbre BPT construit, l'information d'occlusion est utilisée pour effectuer un élagage. L'élagage permet d'obtenir une partition de segmentation. Cette étape est détaillée à la sous-section 1.5.3.

1.5.1.1 Mesure de similarité entre deux régions

La mesure de similarité entre deux régions adjacentes prend en compte l'information couleur, de mouvement ainsi que la forme des deux régions. Elle s'écrit sous forme :

$$Dist(R_1,R_2) = D_{aire} \cdot \left[\alpha \cdot D_{couleur+mouvement} + (1-\alpha) \cdot D_{contour}\right] \tag{1.15}$$

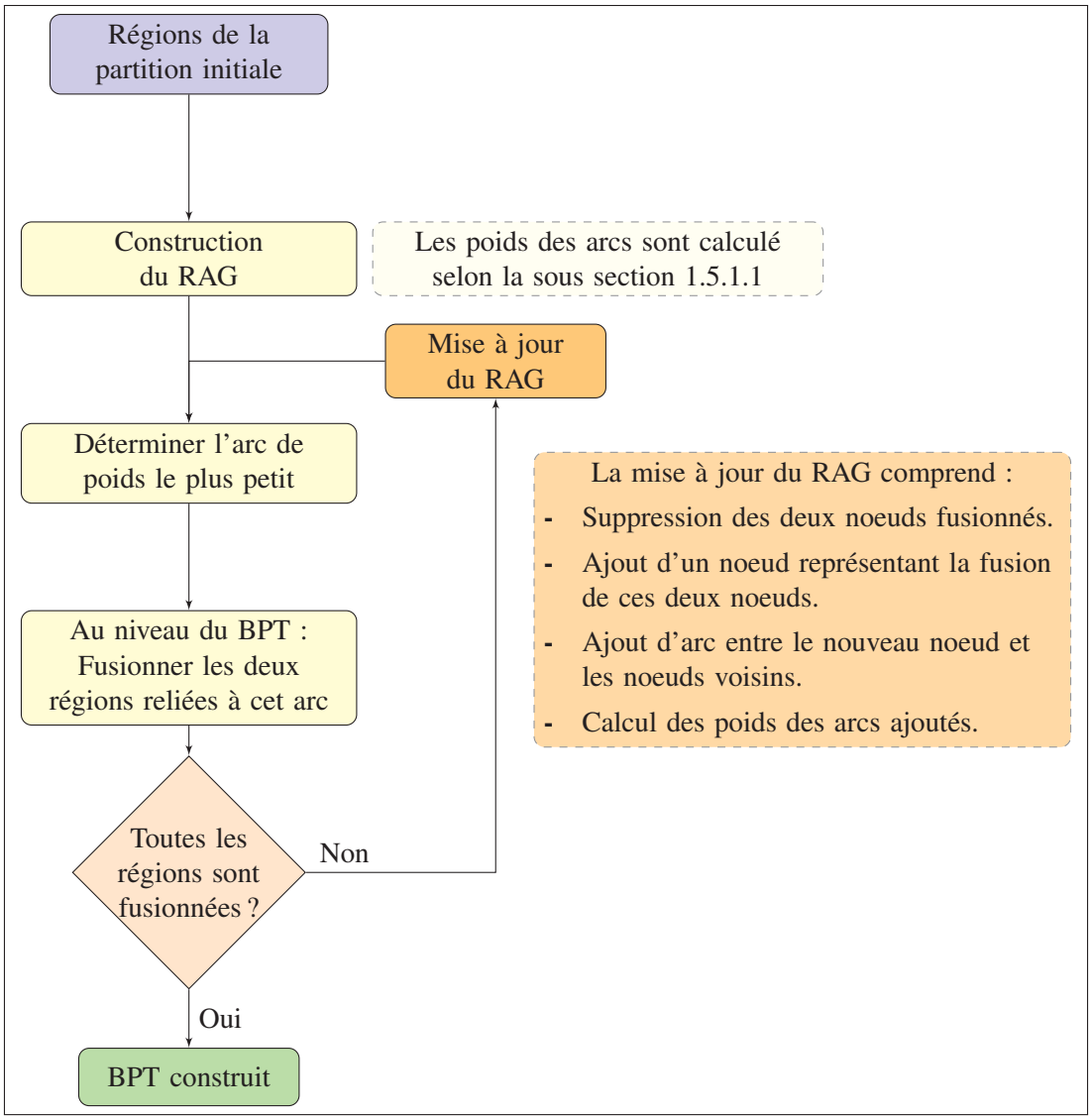


Figure 1.11 Étapes de la construction du BPT

Avec D_{aire} la distance qui considère la superficie des deux régions R_1 et R_2, $D_{contour}$ la distance qui dépend de la forme des deux régions, $D_{couleur+mouvement}$ celle qui tient compte des informations de couleur et mouvement et α est un coefficient compris entre 0 et 1 permettant de pondérer les poids des distances $D_{couleur+mouvement}$ et $D_{contour}$.

Calcul de D_{aire}

Soient A_1 et A_2 les aires respectives des deux régions R_1 et R_2, la distance d'aire s'écrit sous la forme :

$$D_{aire} = \log(1 + min(A_1, A_2))$$

Pondérer la distance $Dist(R_1, R_2)$ par D_{aire} permet d'encourager la fusion des régions de petite taille, et favorise l'équilibre du BPT construit. En effet D_{aire} est proportionnelle à la surface de la plus petite des deux régions, mais selon une échelle logarithmique afin que l'effet des distances $D_{couleur+mouvement}$ et $D_{contour}$ restent prépondérant.

Calcul de $D_{contour}$

Soient P_1 et P_2 les périmètres respectifs des deux régions R_1 et R_2, et P_{1-2} la longueur du contour commun des deux régions. La distance de contour s'écrit sous la forme :

$$D_{contour} = \max\left(0, \frac{\min(P_1, P_2) - 2 \cdot P_{1-2}}{\max(P_1, P_2)}\right) \tag{1.16}$$

$D_{contour}$ permet de favoriser la fusion des régions ayant un large périmètre commun. Plus du poids est donné au terme $D_{countour}$ (c.à.d : Plus le coefficient α est petit dans l'équation 1.15), plus les régions fusionnées seront convexes.

Calcul de $D_{couleur}$

Afin de mesurer la similarité en couleur entre deux régions, un histogramme de couleur tridimensionnel (dans l'espace couleur CIELAB) de chaque région est calculé par application de l'algorithme des K-moyennes. Bien que plus coûteux en calcul, l'utilisation d'un histogramme tridimensionnel permet une représentation plus fidèle de la région que l'utilisation de trois histogrammes mono-dimensionnels. En effet, les histogrammes mono-dimensionnels n'exploitent pas les corrélations entre dimensions. La figure 1.12 expose un cas où il est plus avantageux d'utiliser un histogramme multi-dimensionnel (l'espace RGB est utilisé pour la figure à des fins

d'illustration). Les images 1 et 2 possèdent des histogrammes mono-dimensionnels identiques, bien qu'il soit clair qu'aucune couleur n'est commune aux deux images.

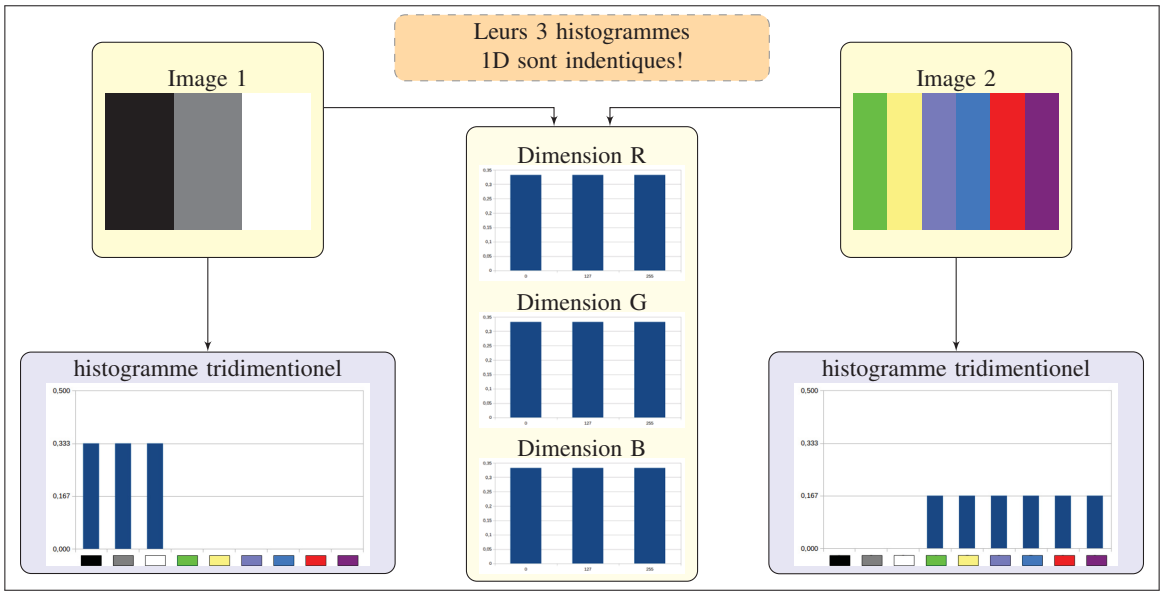


Figure 1.12 Illustration de l'avantage des histogrammes de couleur tridimensionnels

Une fois ces histogrammes obtenus, la distance earth mover (*earth mover's distance*, EMD) introduite par Levina & Bickel (2001) est utilisée. L'EMD permet de quantifier la distance entre deux histogrammes en mesurant le coût de transformation d'un histogramme à l'autre, elle est calculée par résolution du problème de programmation linéaire suivant :

Soient **h** et **g** deux histogrammes, contenant respectivement N_h et N_g intervalles. Chaque intervalle i de **h** (respectivement **g**) a une probabilité h_i (respectivement g_i). La distance EMD entre **h** et **g**, est alors définie comme :

$$EMD(\mathbf{h},\mathbf{g}) = \min_{f_{ij}} \frac{\sum_{i=1}^{N_h}\sum_{j=1}^{N_g} c_{ij} f_{ij}}{\sum_{i=1}^{N_h}\sum_{j=1}^{N_g} f_{ij}} \tag{1.17}$$

Avec

$$f_{ij} \geq 0 \quad ; \quad \sum_{i=1}^{N_h} f_{ij} \leq g_i, \quad \forall j \quad ; \quad \sum_{j=1}^{N_g} f_{ij} \leq h_i, \quad \forall i \tag{1.18}$$

Afin que la distance *EMD* soit normalisée, entre 0 (pour deux histogrammes identiques) et 1 (pour deux histogrammes complètement différents) les coefficients c_{ij} sont définis comme suit :

$$c_{ij} = \left(1 - exp^{-\frac{\Delta_{ij}}{\gamma_c}}\right) \tag{1.19}$$

Avec γ_c fixé à 14 pour le calcul de la distance de couleur et Δ_{ij} la distance euclidienne entre le $i^{ième}$ centre d'intervalle de **h** et le $j^{ième}$ centre d'intervalle de **g**.

Calcul de $D_{mouvement}$

La distance de mouvement considère le mouvement avant $\mathbf{w}_{t\to(t+1)}$ et le mouvement arrière $\mathbf{w}_{t\to(t-1)}$:

$$D_{mouvement} = D_{m(t\to(t+1))} + D_{m(t\to(t-1))} \tag{1.20}$$

Les termes $D_{m(t\to q)}$ avec $q = t \pm 1$, se calculent de façon similaire à la distance couleur, en considérant le flot optique $\mathbf{w}_{t\to q}$ à la place de la couleur. Les histogrammes sont donc bidimensionnels. À la seule différence que le terme γ_c est remplacé par γ_m dans l'équation 1.19 afin d'assurer la normalisation de l'EMD.

γ_m est défini par

$$\gamma_m = 0.25(m_{max} - m_{min})$$

où

$$m_{max} = \max_{q,p} |\mathbf{w}_{t\to q}(p)| \qquad m_{min} = \min_{q,p} |\mathbf{w}_{t\to q}(p)|$$

Avec $q = t+1$ ou $q = t-1$ et p tous les pixels de la trame traitée.

$D_{mouvement}$ et $D_{couleur}$ sont combinés pour obtenir $D_{mouvement+couleur}$, Palou (2013); Salembier & Palou (2014) ne précise pas s'il s'agit d'une simple addition ou d'une combinaison plus élaborée.

1.5.2 Calcul des occlusions

Pour le calcul des occlusions, Salembier & Palou (2014) exploitent le principe qu'en absence d'occlusions entre deux trames, le flot optique devrait être une bijection. Ainsi quand deux pixels déplacés par leurs flots optiques respectifs arrivent au même endroit, ils en déduisent que l'un de ces deux pixels est un pixel caché :
Si pour deux pixels $\mathbf{x}_a$ et $\mathbf{x}_b$ avec $\mathbf{x}_a \neq \mathbf{x}_b$, l'équation ci-dessous est vérifiée :

$$\mathbf{x}_a + \mathbf{w}_{t\to(t+1)}(\mathbf{x}_a) = \mathbf{x}_b + \mathbf{w}_{t\to(t+1)}(\mathbf{x}_b) = \mathbf{x}_m \tag{1.21}$$

Alors, l'un des deux pixels ($\mathbf{x}_a$ et $\mathbf{x}_b$) est caché. Par la suite, pour distinguer lequel des deux est caché, le voisinage de couleur de chacun des deux pixels est utilisé pour calculer la distance suivante :

$$D(\mathbf{x}_p, \mathbf{x}_m) = \sum_{d\in\Gamma} [I_{t+1}(\mathbf{x}_m + d) - I_t(\mathbf{x}_p + d)]^2 \tag{1.22}$$

Avec $\mathbf{x}_p = \mathbf{x}_a$ ou $\mathbf{x}_b$, et Γ un voisinage de 5×5 pixels.
Le pixel avec le plus grand $D(\mathbf{x}_p, \mathbf{x}_m)$ est considéré caché.

Une fois un pixel $\mathbf{x}_c$ est détecté caché, la détection du pixel qui le cache nécessite un flot optique fiable au point $\mathbf{x}_c$. Pour l'obtenir, le flot optique au niveau région est estimé, selon un modèle quadratique. Ce calcul s'effectue de la manière suivante :
Soit $\mathbf{w}_{t\to t+1}(x,y) = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix}$ le flot optique entre la trame t et $t+1$ au pixel $\mathbf{x} = (x,y)$ et soit R une région de l'image. Le flot optique basée région $\widetilde{\mathbf{w}}^R_{t\to t+1}(x,y) = \begin{pmatrix} \widetilde{u}^R(x,y) \\ \widetilde{v}^R(x,y) \end{pmatrix}$ est exprimé de la façon suivante :

$$\begin{aligned} \widetilde{u}^R(x,y) &= a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy \\ \widetilde{v}^R(x,y) &= a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2 \end{aligned} \tag{1.23}$$

Avec $(x,y) \in R$. L'estimation des paramètres $a_i, i \in \{1,\ldots,8\}$, nécessite l'application d'un algorithme de régression robuste, comme les moindres carrés récursifs. Après l'obtention du

modèle du flot optique basé région, le pixel cachant $\mathbf{x}_o$ d'un pixel caché $\mathbf{x}_c$ est défini selon l'équation :

$$\mathbf{x}_o = \mathbf{x}_c + \widetilde{\mathbf{w}}^R_{t\to t+1}(\mathbf{x}_c) + \mathbf{w}_{t+1\to t}(\mathbf{x}_c + \widetilde{\mathbf{w}}^R_{t\to t+1}(\mathbf{x}_c)) \tag{1.24}$$

Cette approche présente des limitations :

- L'effet de régularisation du flot optique introduit plusieurs faux positifs.
- La distinction du pixel caché parmi la paire $(\mathbf{x}_a, \mathbf{x}_b)$ s'effectue à l'aide du voisinage couleur, cela entraîne des erreurs même dans le cas d'une estimation de mouvement parfaite.
- Le calcul du flot optique estimé au niveau région est nécessaire.

Dans (Palou, 2013), une variante du calcul des occlusions est proposée, cette variante élimine l'utilisation de l'information de couleur pour la détection des pixels cachés, et se base plutôt sur l'information de segmentation. En effet, dans cette approche un pixel est considéré caché si déplacé par le flot optique estimé de sa région, il atterrit dans une autre région, selon l'équation :

$$\Lambda(\mathbf{x}_c) \neq \Lambda(\mathbf{x}_c + \widetilde{\mathbf{w}}^R_{t\to t+1}(\mathbf{x}_c) + \mathbf{w}_{t+1\to t}(\mathbf{x}_c + \widetilde{\mathbf{w}}^R_{t\to t+1}(\mathbf{x}_c))) \tag{1.25}$$

Avec Λ un opérateur associant chaque pixel à la région de la segmentation considérée. Cette variante permet de dépasser les deux premières limitations citées ci-haut, mais nécessite toujours l'estimation du flot optique basée région ainsi que la connaissance préalable de régions de segmentations. Afin de remédier à ces limitations, une autre méthode de calcul des occlusions est proposée dans la présente étude. Cette méthode se base sur le flot optique, mais sans faire intervenir l'information couleur, et ne nécessite pas l'estimation du flot optique par région. Cette méthode est détaillée au chapitre 2, section 2.3.

1.5.3 Élagages

Une fois l'arbre BPT obtenu, Salembier & Palou (2014) l'élaguent en deux étapes, pour obtenir une segmentation.

Soit R_i une région du BPT et R_g (respectivement R_d) sa région enfant gauche (respectivement enfant droit). L'élagage consiste à décider pour chaque région R_i, s'il est préférable que la partition voulue comprenne les régions R_g et R_d de façon séparée, ou bien de façon fusionnée (R_i). Ainsi, pour effectuer un élagage, une fonction d'énergie de région est définie $E(R)$. Puis l'arbre est parcouru en commençant par les feuilles. À chaque région R_i visitée, il y a élagage (les deux enfants R_g et R_d sont supprimés) si :

$$E(R_i) < E(R_g) + E(R_d) \tag{1.26}$$

Les deux élagages se différencient par leurs fonctions énergie. Comme détaillé ci-dessous :

1.5.3.1 Premier élagage

Le premier élagage utilise l'erreur qu'introduit l'estimation du flot optique par région selon l'équation 1.23. En effet, la fonction énergie du premier élagage est :

$$E(R_i) = \sum_{q=t\pm1} \sum_{(x,y)\in R_i} \left| \mathbf{w}^R_{t\to q}(x,y) - \widetilde{\mathbf{w}}^R_{t\to q}(x,y) \right| + \lambda_f \tag{1.27}$$

Avec $\lambda_f = 4\times10^3$. Ce paramètre λ_f permet d'éviter la sur-segmentation. Cet élagage nécessite l'estimation du flot optique par région pour chaque région traitée.

1.5.3.2 Second élagage

Le second élagage utilise, quant à lui, les paires de pixels $(\mathbf{x}_c, \mathbf{x}_o)$ calculées selon l'approche présentée à la section 1.5.2. Sa fonction énergie est :

$$E(R_i) = \sum_{(\mathbf{x}_c,\mathbf{x}_o)\in R} \frac{1}{N_o} + \lambda_o \tag{1.28}$$

Avec N_o le nombre total de paires (x_c, x_o) détectées et $\lambda_o = 4\times10^{-3}$. Ce second élagage assure que la partition finale obtenue contient des régions qui pourront être ordonnées entre elles.

Ce sont les feuilles du BPT obtenu à la suite des deux élagages qui constitue la partition de segmentation P_o qui sera considérée pour la déduction de l'ordre et l'obtention de la carte de profondeur relative.

1.5.4 Déduction de l'ordre global

Une fois la partition P_o obtenue, une dernière étape permet de déduire l'ordre des régions de cette partition. Pour cela, un graphe dirigé $G = (V, E)$ est construit. Les noeuds de ce graphe représentent les régions de la partition P_o, un arc dirigé $e_i = (a, b, p_i)$ relie le noeud a au noeud b s'il y a relation d'occlusions entre les deux régions. Le poids p_i d'un arc p_i représente la probabilité que la région R_a précède la région R_b. En effet, $p_i = N_{ab}/N_o$. Avec N_o le nombre total des paires d'occlusions et N_{ab} le nombre de paires de pixels $(\mathbf{x}_c, \mathbf{x}_o)$ qui vérifient les conditions $\mathbf{x}_c \in R_b$ et $\mathbf{x}_o \in R_a$.
Pour qu'un ordre global des régions de la partition P_o puisse être déduit, il est nécessaire que le graphe G soit acyclique. Pour cela, les arcs de plus faibles poids causant des cycles dans G sont éliminés. Il en résulte un graphe acyclique de profondeur (*depth acyclic graph*, DAG).
Un tri topologique partiel est appliqué au DAG afin d'assigner un ordre à chaque région de la partition P_o.

1.6 Conclusion

La littérature étudiée fait apparaître que de nombreuses recherches ont été menées dans le domaine de la conversion 2D à 3D semi-automatique. Ces recherches couvrent un large éventail de solutions viables, permettant de réduire de manière significative l'intervention humaine nécessaire pour la conversion 2D à 3D. La question de la conversion 2D à 3D automatique est elle aussi largement traitée par la littérature ; seulement cette tâche est plus complexe. En effet, elle comporte un défi de taille qui consiste à extraire des indices de profondeurs à partir de contenu 2D et d'en générer une carte de profondeur pour chaque image traitée. Étant donné la complexité de cette conversion, l'état actuel de la littérature n'offre pas encore de solution

fiable, rapide et valable pour toutes les configurations (mouvement de la caméra, de la scène ou des deux).

Une des méthodes figurant dans la littérature (Salembier & Palou (2014), détaillée à la section 1.5) traite du calcul d'ordre de profondeur. Cette méthode, qui permet de générer des cartes de profondeur relatives, présente de nombreux avantages. Elle utilise l'indice des occlusions dynamiques, qui est fiable et présent dans tous les types de scènes, pour ordonner des régions obtenues par une segmentation basée sur la couleur et le mouvement. Dans cette approche, le calcul des relations d'occlusion s'effectue par la vérification de la cohérence avant-arrière du flot optique. Ce calcul nécessite l'estimation du flot optique par région selon un modèle quadratique en utilisant une méthode de régression itérative robuste des moindres carrées. Cette estimation par région est une étape coûteuse dans l'approche de Salembier & Palou (2014), mais elle est nécessaire pour combler le manque de précision du flot optique utilisé (LDOF).

L'étude de la littérature dans le cadre de l'estimation du flot optique montre que de récentes approches permettent l'obtention de flots optiques ayant des propriétés intéressantes. Ainsi, le flot optique EpicFlow, proposé par Revaud *et al.* (2015), permet la préservation des contours. La préservation des contours présente l'avantage d'éviter la perte de précision au niveau des régions avoisinant les contours. Cette perte est courante dans les approches classiques du calcul du flot optique bien que ce soit dans ces régions que se trouvent les occlusions. La cohérence entre le mouvement en avant et celui en arrière est également une propriété du flot optique utile pour le calcul des occlusions, mais l'EpicFlow ne présente pas cette propriété.

Le chapitre qui suit détaille l'approche proposée par le présent travail. Elle comporte des modifications à l'algorithme de l'EpicFlow, pour permettre le calcul conjoint de flots optiques avants et arrières cohérents. Une méthode de calcul des occlusions basée sur le flot optique ainsi modifié est également détaillée, cette méthode ne nécessite pas l'estimation de flot optique par région. L'estimation du flot optique et des occlusions sont utilisées dans une approche semblable à celle suivie par Salembier & Palou (2014) pour générer automatiquement des cartes

de profondeur relative. À la fin du chapitre, un tableau résume les différences entre l'approche proposée et celle de Salembier & Palou (2014).

CHAPITRE 2

DESCRIPTION DE LA MÉTHODE PROPOSÉE

Ce chapitre présente la méthode proposée. La première section présente un aperçu général de l'approche utilisée et les sections qui suivent détaillent les étapes de cette approche.

2.1 Vue globale

Le but est d'obtenir une carte de profondeur relative des objets contenus dans une trame d'une séquence vidéo. Les informations utilisées sont extraites de cette trame ainsi que de la trame la précédant et de la trame lui succédant dans la séquence. La figure 2.1 illustre les étapes de la méthode proposée. L'approche suivie est inspirée des études Palou (2013) et Salembier & Palou (2014). Elle comprend les étapes suivantes :

- Les mouvements avant et arrière entre la trame traitée et celle qui la suit, ainsi que ceux entre la trame traitée et celle qui la précède sont calculés par flot optique (détaillé à la section 2.2).
- Les occlusions sont calculées à l'aide des flots optiques avant et arrière (détaillés à la section 2.3).
- Une partition initiale, calculée à l'aide de méthodes de superpixels, est utilisée pour réduire la complexité de l'étape de la segmentation (détaillée à la section 2.4).
- Une segmentation, basée sur la couleur et le mouvement, est obtenue par construction d'un arbre de partition binaire et ensuite l'élagage de celui-ci (détaillée aux sections 2.5 et 2.6).
- Les relations d'ordre entre régions segmentées sont déduites par les paires d'occlusions pixel caché/pixel cachant.

L'approche (Palou, 2013; Salembier & Palou, 2014) utilise la méthode LDOF (Brox & Malik, 2011) pour l'estimation du flot optique. Cette méthode est performante, mais elle ne prend pas en compte les contours, ce qui a pour effet de lisser les régions proches des contours. Pour pallier cet inconvénient, elle introduit une étape d'estimation du flot optique au niveau région

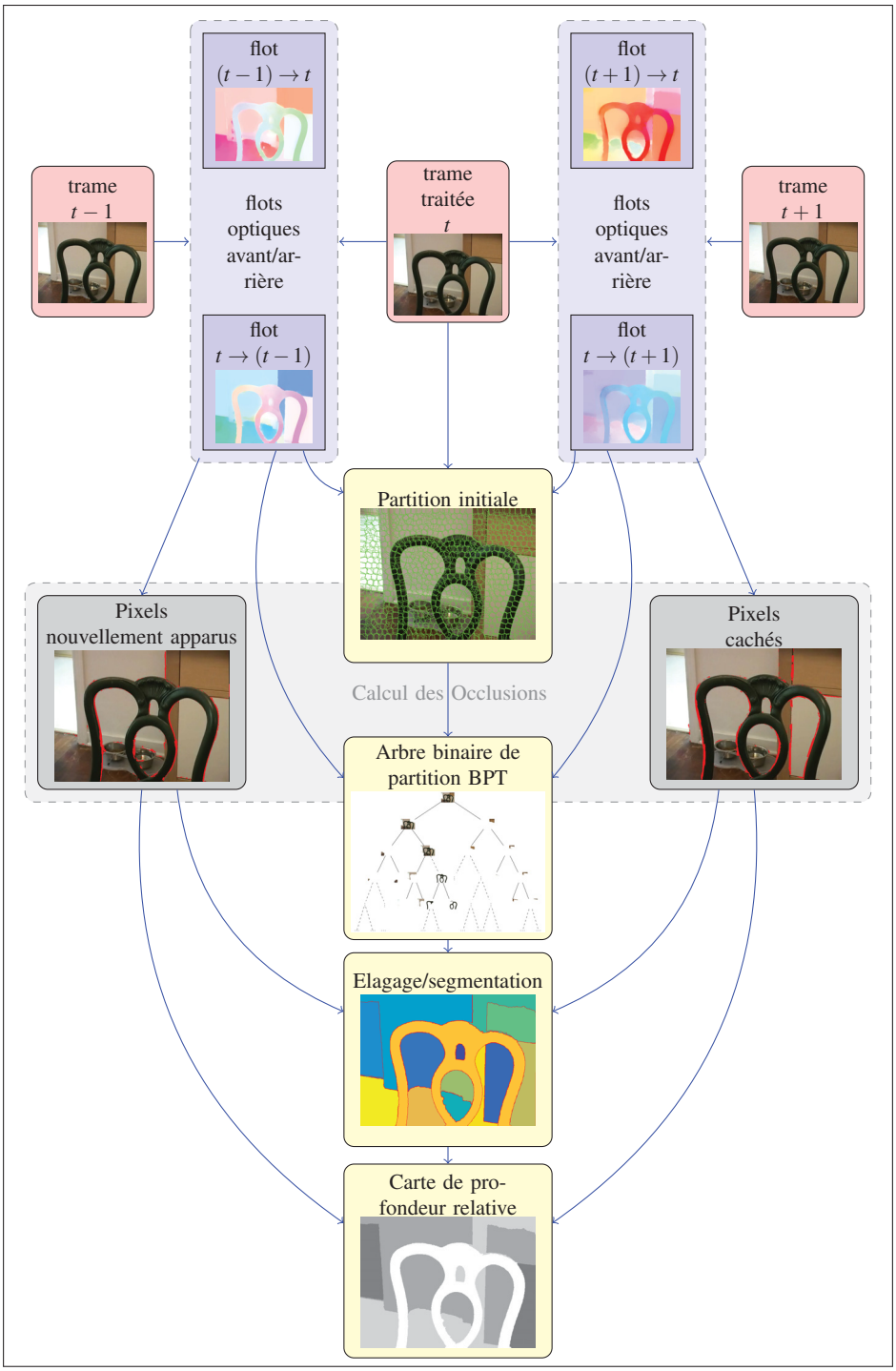


Figure 2.1 Schéma bloc de la méthode suivie

selon un modèle quadratique. La présente étude a choisi d'opter pour le flot optique EpicFlow (Revaud *et al.*, 2015) qui a l'avantage de considérer les contours. Des modifications ont été ap-

portées à la méthode EpicFlow. Ces modifications ont permis de rendre le flot optique cohérent avant/arrière. Grâce à ce flot optique préservant les contours et cohérent avant/arrière, il a été possible de supprimer l'étape de l'estimation du flot optique au niveau région, opération coûteuse en temps et ressources de calcul. La section 2.7 résume les différences entre la méthode proposée et celle de (Palou, 2013; Salembier & Palou, 2014).

2.2 Version modifiée de l'EpicFlow

Pour rendre l'EpicFlow cohérent avant-arrière sans perte de performances, des modifications lui ont été apportées. La figure 2.2 illustre la démarche suivie.

Le calcul des deux flots optiques $\mathbf{w}_{t\rightarrow(t+1)}$ et $\mathbf{w}_{(t+1)\rightarrow t}$ s'effectue de manière conjointe, selon la démarche suivante :

2.2.1 Cohérence avant-arrière des paires de correspondances

L'algorithme de correspondance DeepMatching (Weinzaepfel *et al.*, 2013) ne donne pas les mêmes points de correspondance lorsqu'il fait correspondre I_t avec I_{t+1} ou qu'il fait correspondre I_{t+1} avec I_t. Pour remédier à cet inconvénient, la correspondance est effectuée dans les deux sens. Ensuite, les deux résultats de correspondance sont fusionnés, comme illustré par la figure 2.3. On obtient ainsi un résultat cohérent dans les deux sens et qui contient plus de paires de correspondances.

2.2.2 Cohérence avant-arrière des régions de Voronoï

Afin de favoriser la cohérence avant-arrière, maintenant qu'il y a cohérence au niveau des paires de correspondances, le calcul des régions de Voronoï s'effectue de façon à avoir une cohérence entre régions également.

Le calcul de ses régions remplace la distance géodésique de contours utilisée par la méthode originale de l'EpicFlow par une distance qui considère l'information de contours mais aussi

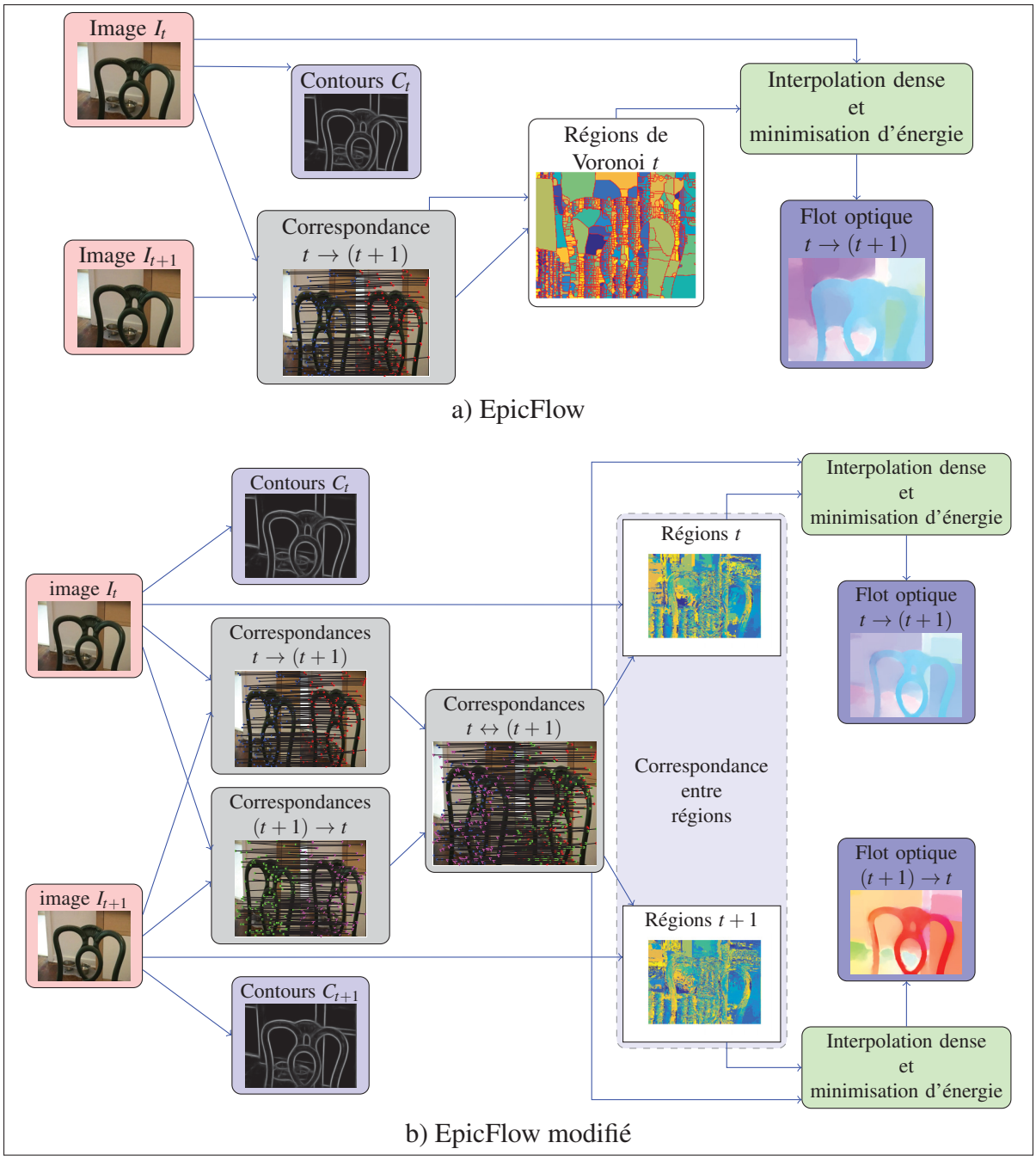


Figure 2.2 Schéma bloc de l'EpicFlow et de l'EpicFlow modifié

de couleur, afin de favoriser la cohérence. En effet, la méthode originale de l'EpicFlow base la création de ces régions uniquement sur la distance géodésique de contours, ce qui a pour effet de créer pour l'image I_{t+1} des régions de formes très différentes de celles de l'image I_t (même

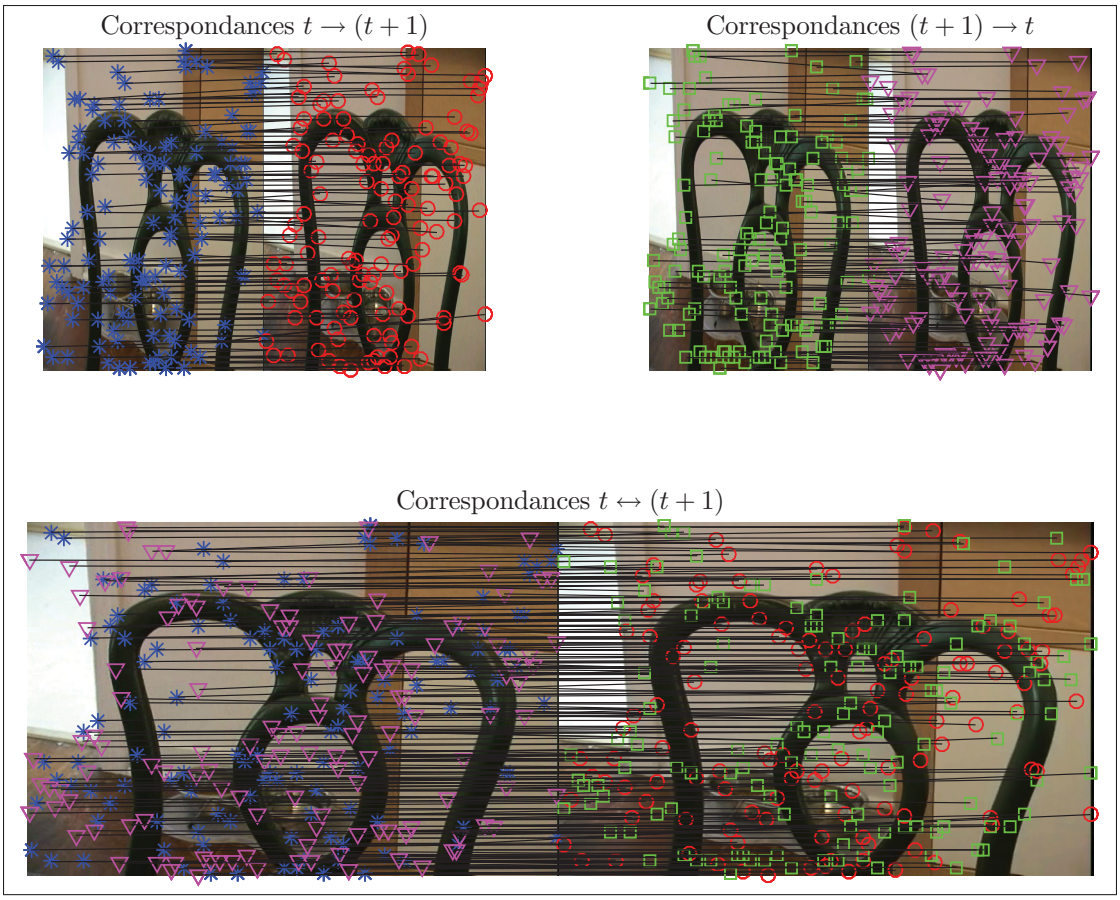


Figure 2.3 Fusion des correspondances $t \rightarrow (t+1)$ et $(t+1) \rightarrow t$

quand les centres des régions, utilisés pour le calcul des régions de I_t et de I_{t+1} proviennent de la même liste de correspondance de point entre ces deux images). Un autre avantage d'introduire l'utilisation de la couleur pour le calcul des régions est que les pixels se trouvant eux-mêmes sur les contours soient affectés à la bonne région. Une mesure qui n'est pas considérée dans le calcul de l'EpicFlow (Revaud *et al.*, 2015).

L'algorithme 2.1 détaille le nouveau calcul de ces régions. Celui-ci est inspiré de la méthode de calcul des superpixels de simple regroupement linéaire itératif (*simple linear iterative clustering*, SLIC) (Achanta *et al.*, 2012). L'idée est qu'au lieu d'obtenir des régions de superpixels dont les centres ont été initialisés selon une grille régulière sur l'image, ces centres $P_k = (x_k, y_k)$ sont initialisés par les points mis en correspondance. La distance utilisée pour décider à quel centre P_k un pixel $P = (x, y)$ est le plus proche est adaptée afin de considérer la couleur et l'in-

formation de contour. Au lieu de la distance couleur et de la distance Euclidienne originalement utilisée par les superpixels SLIC. La distance utilisée s'exprime de la façon suivante :

$$D_r(P_k,P) = D_{lab}(P_k,P) + D_{contour}(P_k,P) \tag{2.1}$$

Avec

$$D_{lab}(P_k,P) = (I_l(Pk) - I_l(P))^2 + (I_a(Pk) - I_a(P))^2 + (I_b(Pk) - I_b(P))^2 \tag{2.2}$$

I_l, I_a,I_b étant respectivement les trois composantes de couleur de l'image dans l'espace couleur CIELAB.

$$D_{contour}(P_k,P) = \left[\sum_{P_i \in L} C(P_i)\right]^2 \tag{2.3}$$

L étant le segment reliant en ligne droite P à P_k et C l'image de contour de I calculé par l'algorithme SED.

2.2.3 Cohérence avant-arrière des graphes d'adjacence

Le graphe d'adjacence G_t est construit pour les régions de l'image t. Le graphe de la seconde image est considéré comme identique à celui de la première $G_{t+1} = G_t$, cette approximation assure la cohérence, avec une moindre perte de performance du calcul du flot optique (la section 3.1 présente une évaluation du flot optique obtenu).

Grâce au flot optique cohérent avant-arrière, les occlusions peuvent être calculées sans avoir recours à un calcul de flot optique estimé par région. La section 2.3 détaille la procédure proposée pour l'estimation de ces relations d'occlusion.

2.3 Estimation des occlusions

L'estimation de l'ordre repose principalement sur la détection des occlusions. En effet, leur calcul est un point clé. La méthode proposée se base sur les trois principes suivants :

Algorithme 2.1 Calcul des régions nécessaires à l'estimation du flot optique

Données : N points mis en correspondances $P_k = (x_k, y_k)$,
I l'image couleur, C l'image contour
Résultat : R l'image des étiquettes des régions

```
1  début
2      D un tableau de même taille que l'image,
3      initialisé à la valeur maximale
4      k ← 1
5      tant que k ≤ N faire
6          x1 ← x_k − offset
7          x2 ← x_k + offset
8          y1 ← y_k − offset
9          y2 ← y_k + offset
10         x ← x_1
11         tant que x ≤ x_2 faire
12             y ← y_1
13             tant que y ≤ y_2 faire
14                 d ← Distance_m((x,y),(x_k,y_k))si d < D(x,y) alors
15                     D(x,y) ← d
16                     R(x,y) ← k
17                 fin
18                 y ← y + 1
19             fin
20             x ← x + 1
21         fin
22         k ← k + 1
23     fin
24     retourner R
25 fin
```

- Il n'y a pas de compatibilité de mouvement avant-arrière lorsque des pixels cachés ou nouvellement apparus sont présents.
- Les pixels cachés / nouvellement apparus se trouvent proches des frontières des objets (en admettant que les objets sont rigides).
- On peut distinguer les pixels cachés de ceux qui apparaissent en vérifiant le comportement du voisinage.

La procédure proposée pour l'estimation de ces relations d'occlusion est résumée à la figure 2.4.

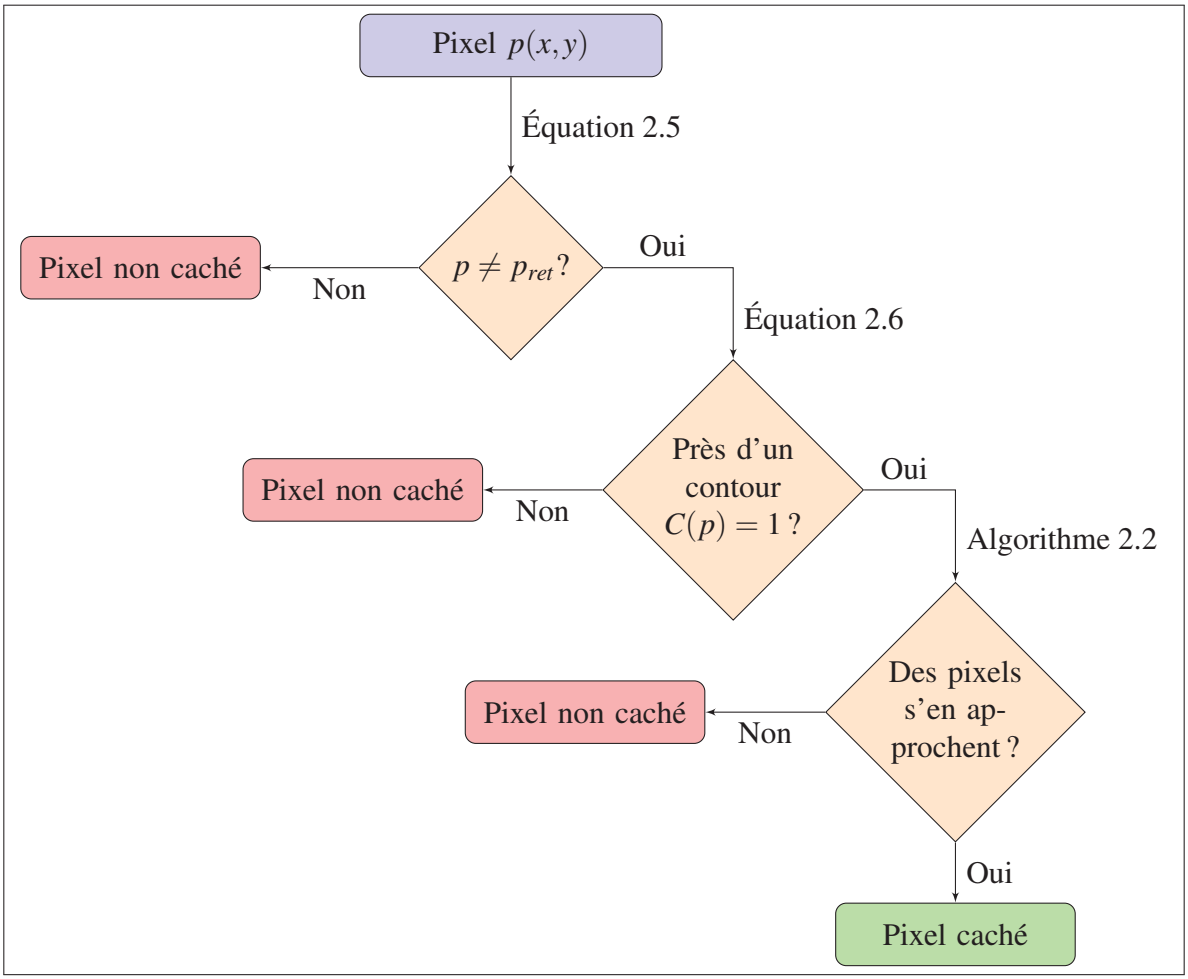


Figure 2.4 Diagramme récapitulatif de la détection d'un pixel caché

Soit $I_t(x,y)$ la trame actuelle traitée et I_{t+1} la trame suivante. $\mathbf{w}_{t\to(t+1)}$ et $\mathbf{w}_{(t+1)\to t}$ désignent respectivement les flots optiques avant et arrière. Un pixel $p = (x,y)$ est potentiellement caché si :

$$p \neq p_{ret} \tag{2.4}$$

Avec

$$\begin{cases} p_{t+1} = p + \mathbf{w}_{t\to(t+1)}(p) \\ p_{ret} = p_{t+1} + \mathbf{w}_{(t+1)\to t}(p_{t+1}) \end{cases} \tag{2.5}$$

Soit p_1 le pixel, de l'image courante, vérifiant cette condition.
Il est ensuite vérifié que le pixel p_1 est proche d'un contour. Les contours C_t précédemment calculés par la méthode SED (Dollár & Zitnick, 2015) sont utilisés à cet effet, selon la condition suivante :

$$M_t(p_1) == 1 \tag{2.6}$$

$M_t(p_1)$ étant l'image binaire dilatée dans un voisinage carré 3×3 de C_t^*, avec :

$$C_t^*(p) = \begin{cases} 1 & \textbf{Si } C_t(p) > C_{seuil} \\ 0 & \textbf{Sinon} \end{cases} \tag{2.7}$$

Avec C_{seuil} fixé de façon empirique à 0.15.

Soit p_2 le pixel vérifiant ces conditions.

Le pixel p_2 est proche d'un contour et le flot optique y est incohérent avant/arrière. Afin de s'assurer que p_2 est un pixel caché est non pas un pixel nouvellement apparu, il est vérifié que les pixels à son voisinage déplacés par le $\mathbf{w}_{t\to(t+1)}$ s'approchent de p_{t+1}. Les pixels voisins considérés sont ceux sur la direction normale au gradient du contour à la position p_2. N_v pixels sont considérés de chacun des deux cotés du pixel p_2 selon cette direction (N_v fixé à 10 pour cette étude). Afin qu'un pixel soit considéré caché, il faut que plus de N_p pixels voisins issus du même coté de la normale, déplacés par le $\mathbf{w}_{t\to(t+1)}$, s'approchent de p_{t+1}. N_p est un nombre vérifiant la propriété $0 < N_p \leq N_v$ (fixé à 8 pour cette étude). L'algorithme 2.2 détaille cette démarche.

Soit p_c un pixel détecté caché, son pixel cachant est $p_o = p_c + \mathbf{w}_{t\to(t+1)}(p_c) + \mathbf{w}_{(t+1)\to t}(p_c + \mathbf{w}_{t\to(t+1)}(p_c))$. L'ensemble des paires (p_c, p_o) détectées forment l'ensemble $\mathscr{L}_{t+1}$ des occlusions détectées entre la trame t et $t+1$.

Algorithme 2.2 Détection d'un pixel caché

Données : $p_2 = (i, j)$ pixel potentiellement caché, α direction du gradient C_t en p_2,$\mathbf{w}_{t\to(t+1)}$
Résultat : $cache = true$ si le pixel est caché ; Sinon $cache = false$

début
  $cache \leftarrow false$
  $(ii, jj) \leftarrow (i, j) + \mathbf{w}_{t\to(t+1)}(i, j)$
  pour $s \in \{1, -1\}$ **faire**
    $N \leftarrow 0$
    $k \leftarrow 1$
    pour $k \leq N_v$ **faire**
      $i_k \leftarrow round(i + k \cdot s \cdot \sin(\alpha))$
      $j_k \leftarrow round(j + k \cdot s \cdot \cos(\alpha))$
      $(ii_k, jj_k) \leftarrow (i_k, j_k) + \mathbf{w}_{t\to(t+1)}(i_k, j_k)$
      $d_t \leftarrow dist((i, j), (i_k, j_k))$
      $d_{t+1} \leftarrow dist((ii, jj), (ii_k, jj_k))$
      si $d_{t+1} < d_t$ **alors**
        $N \leftarrow N + 1$
      fin
    fin
    si $N > N_p$ **alors**
      $cache \leftarrow true$
    fin
  fin
  retourner cache
fin

Le calcul des pixels nouvellement apparus à la trame I_t par rapport à la trame I_{t-1} s'effectue de façon analogue à celle des pixels cachés à la trame I_t par rapport à la trame I_{t+1}. Il suffit de remplacer $t+1$ par $t-1$ dans les équations précédentes pour former $\mathscr{L}_{t-1}$. En effet, les pixels nouvellement apparus à la trame I_t depuis I_{t-1} sont les pixels cachés en I_{t-1} par rapport à la trame I_t.

2.4 Partition initiale

Au lieu de commencer la construction de l'arbre BPT au niveau pixels, l'idée est de déterminer une partition initiale constituée de régions suffisamment petites, afin de s'assurer que la trame traitée est sursegmentée. Ainsi, utiliser une partition initiale réduit la complexité de calcul pour traiter une trame, tout en ayant un effet moindre sur le résultat final.

La méthode de calcul des superpixels simple regroupement linéaire itératif (*simple linear iterative clustering*, SLIC) (Achanta *et al.*, 2012) a été utilisée pour générer la partition initiale. Cette méthode permet de créer des groupes de pixels basés sur leur similarité en couleur (Dans l'espace de couleur CIELAB) et leurs proximités sur l'image.

Dans le cadre de ce travail, la partition initiale est obtenue en effectuant l'intersection des partitions des superpixels sur la trame couleur t ainsi que le flot $\mathbf{w}_{t\rightarrow(t-1)}$ et le flot $\mathbf{w}_{t\rightarrow(t+1)}$ convertis en images de couleur [1] comme l'illustre la figure 2.5.

L'obtention de la segmentation initiale par intersection de régions de mouvements et de couleur, a pour coût d'introduire des régions de petites tailles, mais celles-ci assurent que des pixels appartenant à deux régions d'ordre de profondeur distincts ne soient regroupés dans un seul superpixel et nuirait, de ce fait, à la déduction d'ordre de profondeur.

2.5 Construction de l'arbre de partition binaire

La même approche que Palou (2013); Salembier & Palou (2014) (voir section 1.5.1 pour une description détaillée) a été suivie.

La figure 2.6 illustre la composition d'un arbre BPT.

Le calcul de $D_{couleur+mouvement}$ non précisé par Palou (2013); Salembier & Palou (2014) a été effectuée de la façon suivante.

1. Code de conversion disponible à http://vision.middlebury.edu/flow/data/

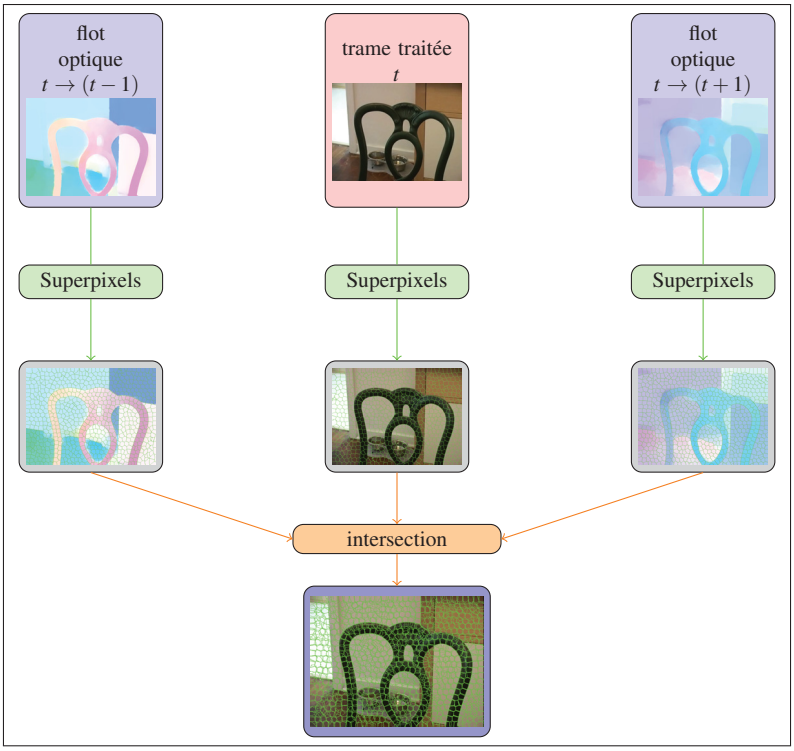


Figure 2.5 Schéma bloc de l'obtention de la partition initiale

Soit $D_{couleur}$ et $D_{mouvement}$ respectivement les distances de couleur et de mouvement entre les deux régions R_1 et R_2. La distance $D_{couleur+mouvement}$ s'exprime sous la forme :

$$D_{couleur+mouvement} = 1.2 \times \left(e^{D_{couleur}} - 1\right) + D_{mouvement} \tag{2.8}$$

2.6 Élagage

Contrairement à Salembier & Palou (2014) où l'élagage est effectué en deux étapes et où les décisions d'élagage nécessitent le calcul préalable du flot optique par région (voir section 1.5.3 pour plus de détails), le présent travail effectue l'élagage en une seule étape et la décision d'élagage utilisée a été adaptée pour ne plus nécessiter l'estimation du flot optique au niveau région. En effet, l'approche suivie par le présent travail fixe un nombre maximal de régions à détecter, soit N_{max} ce nombre. Ainsi, il n'y a que les N_{max} dernières régions fusionnées durant

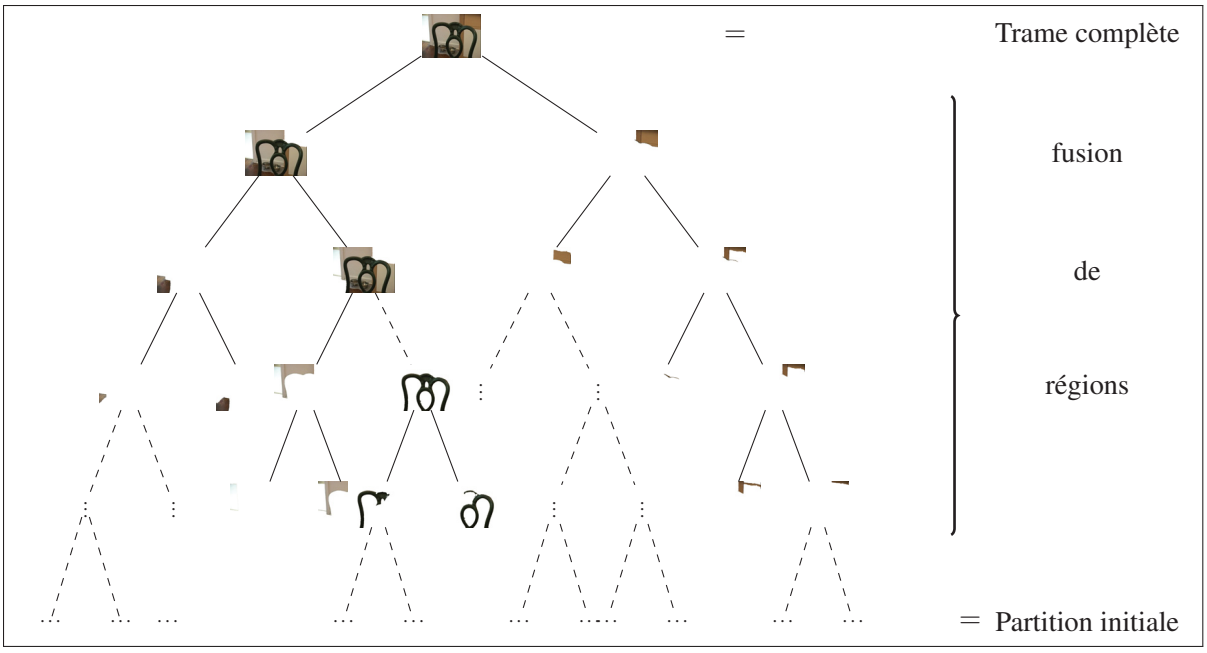


Figure 2.6 Illustration de l'arbre BPT

la construction de BPT qui sont considérées pour l'élagage. Il y a élagage d'une région R_i si

$$E_{t-1}(R_i) \leq N_O \quad \text{et} \quad E_{t+1}(R_i) \leq N_O \tag{2.9}$$

Avec :

$$E_q(R_i) = \sum_{(p_c,p_o)\in\mathscr{L}_q} \Gamma(p_c,p_o) \tag{2.10}$$

Avec $q = t \pm 1$, $\mathscr{L}_q$ l'ensemble des paires de pixels cachés/cachants détectés et :

$$\Gamma(p_c,p_o) = \begin{cases} 1 & \textbf{Si} \quad (p_c \in R_g \text{ et } p_o \in R_d) \text{ ou } (p_c \in R_d \text{ et } p_o \in R_g) \\ 0 & \textbf{Sinon} \end{cases} \tag{2.11}$$

N_O étant un paramètre fixé à 30 pour la présente étude.

Une fois l'élagage effectué, la déduction de l'ordre s'obtient de façon similaire à l'approche de Salembier & Palou (2014), détaillée à la section 1.5.4.

2.7 Résumé des changements par rapport à la méthode de Salembier et Palou

Le tableau 2.1 résume les changements apportés à la méthode présentée par Salembier & Palou (2014). On remarque que l'obtention d'un flot optique préservant les contours et cohérent avant/arrière a permis de faciliter le calcul des occlusions ainsi que la suppression d'un des élagages de la méthode originale.

Tableau 2.1 Tableau récapitulatif des différences entre la méthode originale (Salembier & Palou, 2014) et la méthode proposée

Étape	**Méthode (Salembier & Palou, 2014)**	**Méthode proposée**	**Remarques**
Partition initiale	Niveau pixels	Niveau superpixels basés sur la couleur et le mouvement, voir section 2.4	Avantage : -La réduction de la complexité de la segmentation sans perte de performance.
Calcul du flot optique	Méthode LDOF (Brox & Malik, 2011)	Version modifiée de l'EpicFlow (Revaud *et al.*, 2015), voir section 2.2	Avantages : - Les contours sont préservés. - Cohérence avant/arrière.
Calcul des occlusions	Vérifie si un pixel déplacé par le flot optique région, atterrit dans cette même région une fois re-déplacée par le flot optique arrière	Exploite la cohérence avant/arrière du flot optique ainsi que l'hypothèse admettant que les occlusions ont lieu près des contours, voir section 2.3	La méthode (Salembier & Palou, 2014) nécessite une segmentation et le calcul du flot optique estimé par région.
Construction de l'arbre BPT	Approches quasi similaires		Petites modifications des poids d'arcs du RAG, voir équation 2.8.
Élagage(s)	Élagages en deux étapes	Un seul élagage	Le premier élagage de la méthode (Salembier & Palou, 2014) nécessite le calcul du flot optique estimé par région.
Déduction de l'ordre	Approches similaires		

2.8 Conclusion

La méthode proposée par le présent travail a été exposée dans ce chapitre.

L'obtention d'un flot optique cohérent avant-arrière a permis de faciliter le calcul des relations d'occlusion. Suite à ce résultat, la méthode de Salembier & Palou (2014) a été adaptée pour pouvoir se passer de l'étape de l'estimation du flot optique au niveau région.

Le chapitre suivant expose les résultats obtenus par la méthode proposée.

CHAPITRE 3

RÉSULTATS ET DISCUSSIONS

Le présent chapitre expose les résultats obtenus par la méthode proposée au chapitre précédent. La première section évalue les performances du flot optique proposé. La seconde section traite des résultats de la génération automatique des cartes de profondeur relatives.

3.1 Évaluation du flot optique

À la section 2.2 sont proposées des modifications au flot optique EpicFlow (Revaud *et al.*, 2015) afin de le rendre cohérent avant-arrière. Le flot optique obtenu est évalué sur la base de données de Middlebury présentée à la sous-section suivante en considérant deux métriques d'évaluation détaillées à la section 3.1.2. La dernière sous-section présente une discussion des résultats obtenus.

3.1.1 Base de données Middlebury

La base de données de Middlebury [1] (Baker *et al.*, 2011) a été utilisée pour évaluer le flot optique proposé. La section suivante présente les résultats obtenus pour la partie de cette base de données pour laquelle la vérité terrain est publiquement disponible. Cette partie se compose de huit séquences et sont présentées à la figure 3.1. Chaque séquence contient 8 images (mis à part les séquences Dimetrodon et Venus qui contiennent uniquement deux images). Les résolution des images est de 640×480 pour les séquences Grove2, Grove3, Urban2 et Urban3 ; de 584×388 pour les séquences RubberWhale, Hydrangea, Dimetrodon et de 420×380 pour la séquence Venus. La vérité terrain d'un seul flot optique est disponible par séquence, il s'agit du flot optique de l'image centrale de la séquence, en sens avant $(\mathbf{w}_{I_{centrale}} \rightarrow \mathbf{w}_{I_{centrale+1}})$.

1. disponible au lien http://vision.middlebury.edu/flow/data/

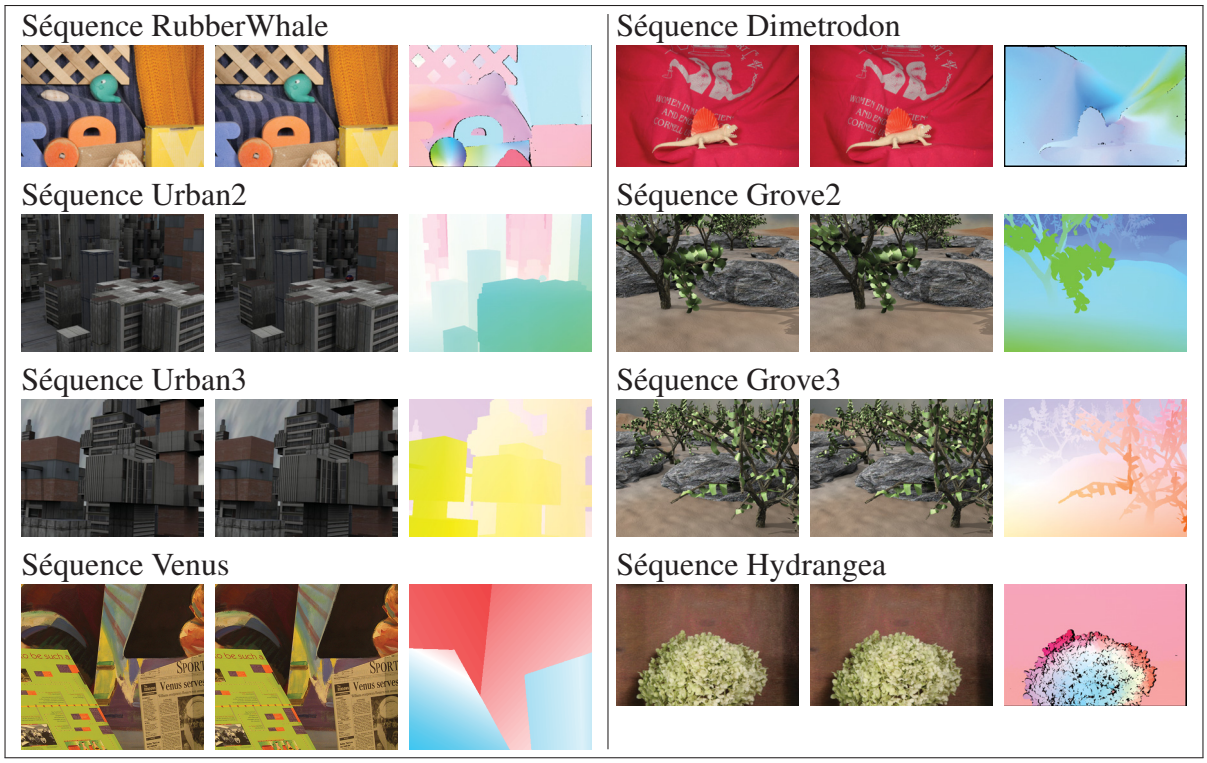


Figure 3.1 Séquences de la base de données Middlebury pour lesquelles la vérité terrain du flot optique est publiquement disponible

3.1.2 Métriques d'évaluation

Deux métriques d'évaluation du flot optique sont couramment utilisées dans la littérature. Il s'agit de l'erreur du point d'arrivée (*end point error*, EPE) et de l'erreur angulaire (*angular error*, AE).

Soit, pour ce qui suit, $\mathbf{w}_v = (u_v, v_v)$ la vérité terrain du flot optique et $\mathbf{w}_e = (u_e, v_e)$ le flot optique estimé.

L'EPE considère la distance entre le point d'arrivée estimé et la vérité terrain du point d'arrivée. Il s'agit d'une erreur moyenne exprimée comme suit :

$$EPE = \frac{1}{N} \sum_{\mathbf{x} \in \Omega'} \sqrt{[u_e(\mathbf{x}) - u_v(\mathbf{x})]^2 + [v_e(\mathbf{x}) - v_v(\mathbf{x})]^2} \tag{3.1}$$

Avec $N = card(\Omega')$ et Ω' l'ensemble de toutes les positions des pixels hormis ceux se trouvant sur la bordure de taille 10 de l'image ainsi que les pixels où la vérité terrain n'est pas définie (ces pixels vérifient l'équation $|\mathbf{w}_v(\mathbf{x})| > 1 \times 10^9$ pour la base de données Middlebury).

Quant à l'EA, c'est aussi une erreur moyenne calculée sur le même ensemble Ω' mais cette distance considère la direction. En effet, c'est l'angle entre le mouvement calculé et la vérité terrain du mouvement qui est pris en compte, de la façon suivante :

$$AE = \frac{1}{N} \sum_{\mathbf{x} \in \Omega'} \cos^{-1} \left(\frac{\overline{\mathbf{w}_v}(\mathbf{x})}{\|\overline{\mathbf{w}_v}(\mathbf{x})\|} \cdot \frac{\overline{\mathbf{w}_e}(\mathbf{x})}{\|\overline{\mathbf{w}_e}(\mathbf{x})\|} \right) \tag{3.2}$$

$$\text{Avec } \overline{\mathbf{w}_v}(\mathbf{x}) = \begin{pmatrix} u_v(\mathbf{x}) \\ v_v(\mathbf{x}) \\ 1 \end{pmatrix} \text{ et } \overline{\mathbf{w}_e}(\mathbf{x}) = \begin{pmatrix} u_e(\mathbf{x}) \\ v_e(\mathbf{x}) \\ 1 \end{pmatrix}$$

3.1.3 Résultats de l'évaluation du flot optique

La méthode de calcul du flot optique proposée à la section 2.2 effectue le calcul conjoint des flots optiques avant ($\mathbf{w}_{t\to(t+1)}$) et arrière ($\mathbf{w}_{(t+1)\to t}$), mais la vérité terrain du mouvement n'est disponible qu'en sens avant ($\mathbf{w}^v_{t\to(t+1)}$). De ce fait, le flot optique est évalué en sens direct où le flot $\mathbf{w}_{t\to(t+1)}$ est un flot avant, mais également en sens inverse où le flot $\mathbf{w}_{t\to(t+1)}$ est un flot arrière. La figure 3.2 illustre ces propos.

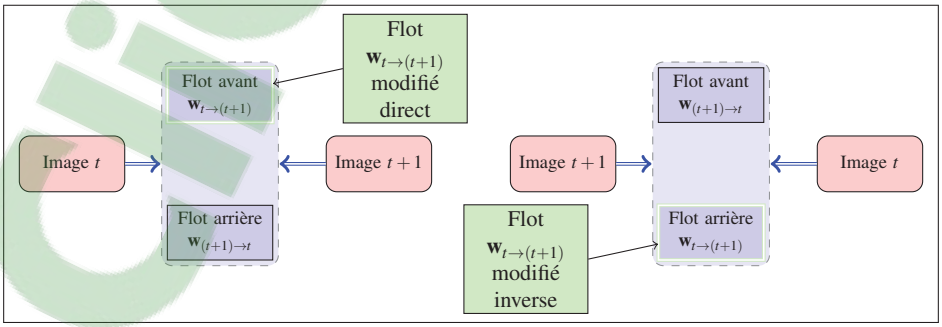


Figure 3.2 Schéma explicatif des nominations des flots optiques modifié direct et modifié inverse

Les tableaux 3.1 et 3.2 contiennent les résultats obtenus pour chacune des images centrales des séquences de la base publique de Middelebury avec la version originale de l'EpicFlow, ainsi que la version modifiée proposée à la section 2.2 en sens direct et inverse. Les deux options d'interpolation proposées par Revaud *et al.* (2015) sont testées.

Tableau 3.1 Résultats obtenus par l'EpicFlow et la version proposée sur la base de test Middelebury selon l'EPE

	Interpolation NW			Interpolation LA		
	Modifiée directe	Modifiée inverse	Originale	Modifiée directe	Modifiée inverse	Originale
RubberWhale	**0.1036**	0.1090	0.1084	**0.1038**	0.1090	0.1084
Urban2	**0.2184**	0.2474	0.2681	**0.2391**	0.2715	0.2485
Urban3	**0.3531**	0.3960	0.5520	**0.4412**	0.4554	0.7591
Venus	**0.2869**	0.2939	0.2959	**0.2876**	0.2968	0.2941
Dimetrodon	**0.0794**	0.0808	0.0808	**0.0794**	0.0808	0.0808
Hydrangea	0.1706	0.1706	**0.1698**	**0.1664**	0.1689	0.1686
Grove2	**0.1402**	0.1438	0.1481	**0.1400**	0.1444	0.1491
Grove3	**0.5691**	0.5807	0.6498	0.5703	**0.5695**	0.6281
Ensemble des séquences	**0.2643**	0.2802	0.3232	**0.2829**	0.2927	0.3517

Tableau 3.2 Résultats obtenus par l'EpicFlow et la version proposée sur la base de test Middelebury selon l'AE

	Interpolation NW			Interpolation LA		
	Modifiée directe	Modifiée inverse	Originale	Modifiée directe	Modifiée inverse	Originale
RubberWhale	**3.4579**	3.6609	3.6513	**3.4573**	3.6632	3.6519
Urban2	**2.2379**	2.3332	2.4246	**2.2574**	2.3206	2.3490
Urban3	**2.6540**	2.8179	3.2487	**2.8367**	2.9672	3.3471
Venus	**3.9752**	4.1769	4.1764	**4.0175**	4.3235	4.2201
Dimetrodon	**1.5666**	1.6042	1.6048	**1.5661**	1.6041	1.6040
Hydrangea	**2.0213**	2.0459	2.0422	**2.0133**	2.0429	2.0417
Grove2	**2.1280**	2.1784	2.2476	**2.1229**	2.1888	2.2639
Grove3	**5.6367**	5.7750	6.2109	5.7418	**5.6940**	6.1553
Ensemble des séquences	**2.9821**	3.0919	3.2675	**3.0351**	3.1096	3.2665

Les résultats de l'EpicFlow (Revaud *et al.*, 2015) ont été obtenus en utilisant le code mis à disposition par les auteurs de la méthode [2] avec les paramètres qu'ils suggèrent d'utiliser pour la base de données Middelebury.

3.1.4 Discussion des résultats du flot optique

On remarque des tableaux 3.1 et 3.2 qu'en moyenne les résultats obtenus par L'EpicFlow modifié sont meilleurs que ceux obtenus par l'EpicFlow (Revaud *et al.*, 2015).
Sur certains exemples l'EpicFlow modifié permet d'obtenir une amélioration assez importante, comme sur l'exemple Urban3. La figure 3 illustre pour cet exemple, la répartition de l'erreur EPE du flot optique $(\mathbf{w}_{I_{centrale}} \rightarrow \mathbf{w}_{I_{centrale+1}})$ de la séquence sur chaque pixel de l'image.

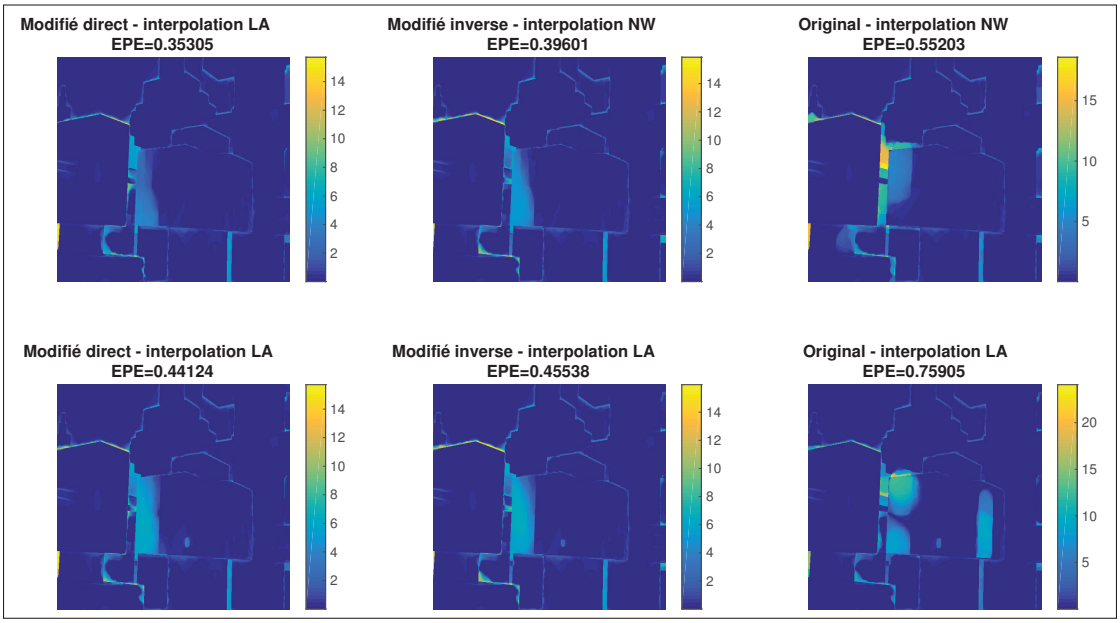


Figure 3.3 Répartition des erreurs EPE sur l'ensemble des pixels de l'image sur l'exemple Urban3 de Middelbury

2. Disponible à https://thoth.inrialpes.fr/src/epicflow/, consulté le 18/07/2017

On peut en conclure que la version du flot optique proposée à la section 2.2 ne détériore pas les performances du flot optique EpicFlow Revaud *et al.* (2015) et apporte globalement de petites améliorations de performance.

De plus, cette méthode apporte la cohérence avant-arrière au flot optique. Comme l'illustre la figure 3.4.

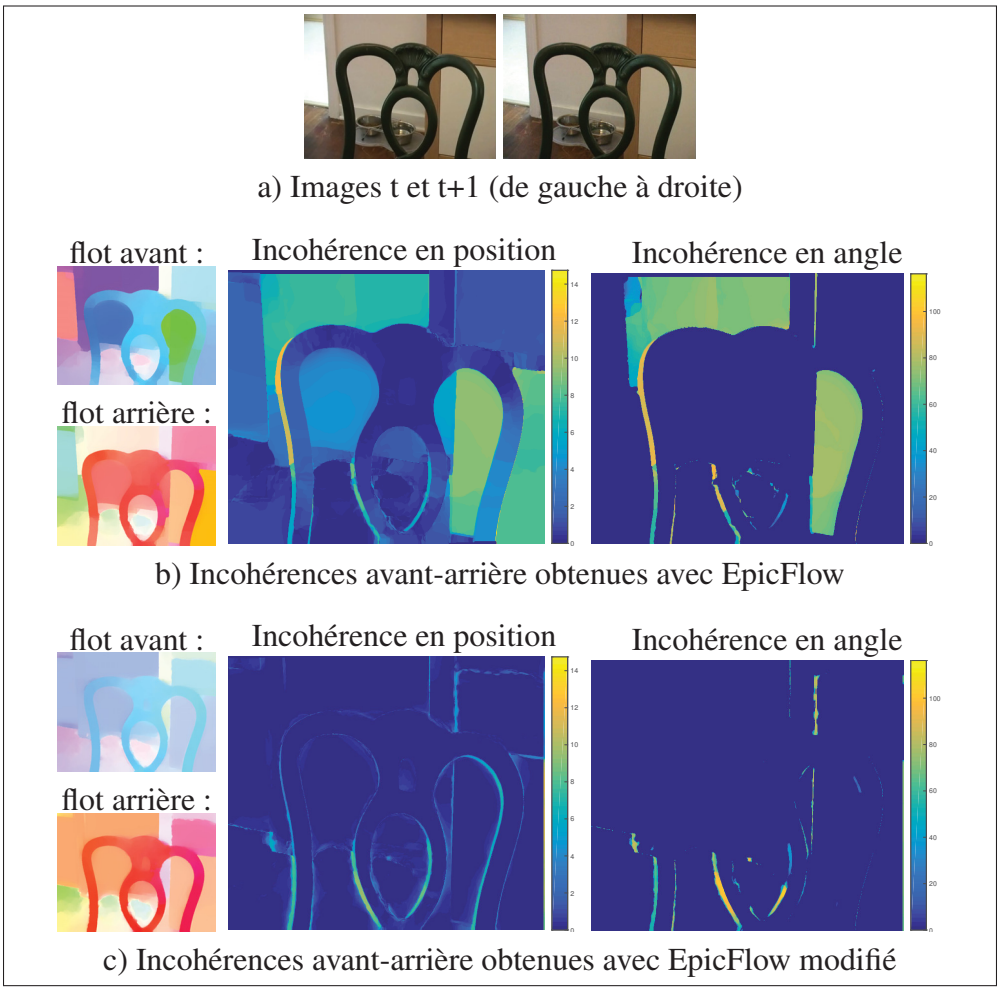


Figure 3.4 Illustration de la cohérence avant-arrière de l'EpicFlow modifié

L'incohérence du mouvement en position est calculé selon l'expression :

$$Incoherence_{position} = \sqrt{[u_{t\to(t+1)}(\mathbf{x}_t) + u_{(t+1)\to t}(\mathbf{x}_{t+1})]^2 + [v_{t\to(t+1)}(\mathbf{x}_t) + v_{(t+1)\to t}(\mathbf{x}_{t+1})]^2}$$

$$Incoherence_{angle} = \cos^{-1}\left(\frac{\overline{\mathbf{w}_{t\to(t+1)}}(\mathbf{x}_t)}{\left\|\overline{\mathbf{w}_{t\to(t+1)}}(\mathbf{x}_t)\right\|} \cdot \frac{-\overline{\mathbf{w}_{(t+1)\to t}}(\mathbf{x}_{t+1})}{\left\|\overline{\mathbf{w}_{(t+1)\to t}}(\mathbf{x}_{t+1})\right\|}\right)$$

Avec $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{w}_{t\to(t+1)}(\mathbf{x}_t)$

Et $\overline{\mathbf{w}_{t\to(t+1)}}(\mathbf{x}_t) = \begin{pmatrix} u_{t\to(t+1)}(\mathbf{x}_t) \\ v_{t\to(t+1)}(\mathbf{x}_t) \\ 1 \end{pmatrix}$ et $\overline{\mathbf{w}_{(t+1)\to t}}(\mathbf{x}_{t+1}) = \begin{pmatrix} u_{(t+1)\to t}(\mathbf{x}_{t+1}) \\ v_{(t+1)\to t}(\mathbf{x}_{t+1}) \\ 1 \end{pmatrix}$

3.2 Évaluation de la déduction d'ordre

Pour évaluer la méthode proposée, deux bases de données ont été utilisées, elles sont présentées à la sous-section 3.2.1. Puis les métriques d'évaluations sont détaillées à la section 3.2.2. Ensuite la sous-section 3.2.3 présente les résultats obtenus. Finalement, la dernière sous-section présente une discussion des résultats obtenus.

3.2.1 Bases de données utilisées

Les deux bases de données utilisées sont les suivantes.

Base de données de Carnegie Mellon

La base de données de Carnegie Mellon (*Carnegie Mellon dataset*, CMU) [3] (Stein & Hebert, 2009) comporte 30 séquences. Elle est souvent utilisée pour l'évaluation de la détection des contours des occlusions. La longueur des séquences varie de 8 à 30 trames, avec une majorité (20/30) séquences de longueur de 20 trames. La vérité terrain de la segmentation ainsi que de l'ordre est disponible pour chaque trame centrale d'une séquence. La méthode proposée est évaluée sur 29 de ces 30 trames centrales. En effet, la séquence *staplers* a été retirée de l'évaluation, car la résolution de la vérité terrain de l'ordre ainsi que de la segmentation ne correspondent pas à la résolution des trames de cette séquence. La figure 3.5 illustre les trames centrales des séquences de cette base de données.

3. Disponible au lien http://www.cs.cmu.edu/~stein/occlusion_data/

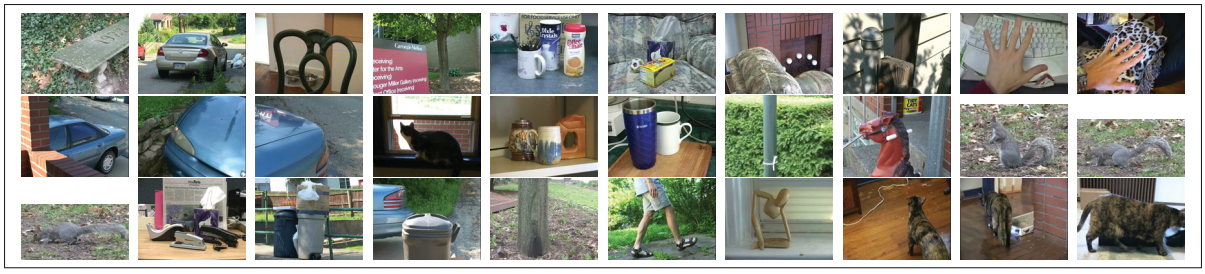


Figure 3.5 Trames de réference des séquence de la base de données CMU

Base de données de Berkeley pour la segmentation vidéo

La base de données de Berkeley pour la segmentation vidéo (*Berkeley video segmentation dataset*, BVSD) [4] (Sundberg *et al.*, 2011) comporte un total de 100 séquences réparties sur deux répertoires : un répertoire test comportant 60 séquences et un répertoire dédié à l'apprentissage contenant 40 séquences. Comme la vérité terrain de la segmentation n'est disponible que pour la trame centrale des séquences du répertoire de test, la méthode n'a été évaluée que pour les 60 trames centrales de ce répertoire. Quant à la vérité terrain de l'ordre, comme celle-ci n'est pas disponible, il a été manuellement attribué à partir de la vérité terrain de la segmentation. La figure 3.6 illustre les trames centrales des 60 séquences du répertoire test.

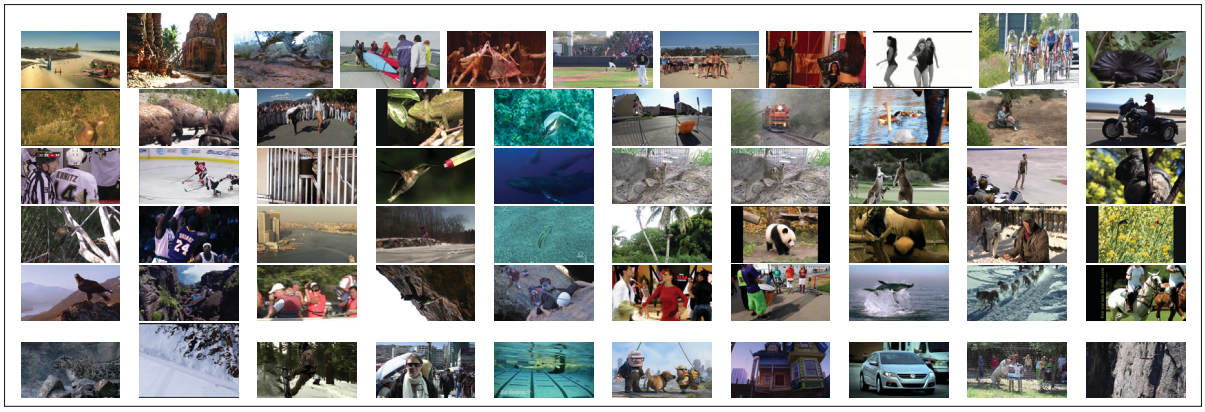


Figure 3.6 Trames de réference des séquences de la base de données BVSD

4. Disponible au lien https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/

3.2.2 Métriques d'évaluation

Pour évaluer les cartes de profondeur relatives obtenues, les métriques de consistance d'ordre local ainsi que l'indice supérieur à l'aléatoire (*over random index*, ORI) proposées par Palou & Salembier (2014) sont utilisées. Elles sont décrites aux sous-sections suivantes.

3.2.2.1 Consistance d'ordre local

Pour évaluer une segmentation, les métriques classiques de précision (*precison*, P), de rappel (*recall*, R) et mesure-F (*F-mesure*, F) sont souvent utilisées dans la littérature.
Pour rappel, ces métriques sont définies en fonction des nombres de détections : vraies positives (*true positives*, TP), fausses positives (*false positives*, FP) et fausses négatives (*false negatives*, FN) comme suit :

$$P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+FN} \tag{3.3}$$

Souvent ces deux métriques sont inversement proportionnelles, plus le rappel sera élevé moins le système sera précis et à l'inverse le système sera précis avec un rappel plus faible. Il s'agit donc de trouver un compromis entre ces deux valeurs. Pour cela, la mesure F est définie, il s'agit de la moyenne géométrique du rappel et de la précision :

$$F = 2 \cdot \frac{P \cdot R}{R+P} \tag{3.4}$$

Quant à la classification, ou dans notre cas l'ordre de profondeur, souvent les résultats sont présentés par des matrices de confusions, permettant de mesurer les performances de la classification. Seulement, comme les régions à ordonner doivent être préalablement détectées, l'évaluation de la classification risque d'être plus performante avec une détection de précision élevée qu'avec un système de faible précision.

Ainsi l'approche de Palou & Salembier (2014) propose des métriques permettant de conjointement évaluer la segmentation et la classification. Pour cela, les deux concepts suivants sont introduits :

- Détection inconsistante (*Inconsistent detection*, ID) : Il s'agit d'une détection correcte ayant été mal classifiée.
- Détection consistante (*consistent detection*, CD) : Il s'agit d'une détection correcte ayant été bien classifiée.

Un détection peut également être vraie négative (*true negative*, TN), fausse détection (*false detection*, FD) ou encore une détection manquée (*missed detection*, MD). Avec ces nouveaux concepts, une matrice de confusion comme représentée par le tableau 3.3 peut être considérée.

Tableau 3.3 Matrice de confusion pour une évaluation conjointe de segmentation et classification

		Détection : $\emptyset$	Détection : 1	
			Classe A	Classe non A
Détection : $\emptyset$		TN	MD	MD
Détection : 1	Classe A	FD	CD	ID
	Classe non A	FD	ID	CD

Ainsi, deux cas extrêmes d'évaluation peuvent être envisagés, l'évaluation qui ne considère que la détection, et celle qui ne considère que la classification :

Système de pure détection

La classification n'est pas prise en compte, les CD et ID sont tous deux considérés corrects. Donc :

- $TP = CD + ID$
- $FP = FD$
- $FN = MD$

Système de détection et classification

La classification est prise en compte, les CD sont considérés corrects et le ID ne le sont pas. Donc :

- $TP = CD$
- $FP = FD + ID$
- $FN = MD + ID$

Pour le système de detection et classification, ID intervient à la fois pour FP et FN. En effet, classifier incorrectement une région signifie, d'une part, détecter une région qui n'existe pas

(d'où la contribution à FP) et signifie, d'autre part, qu'une région présente sur la vérité terrain n'a pas été détectée (d'où la contribution à FN).

Une évaluation entre ces deux systèmes extrêmes peut être considérée, en définissant un paramètre β avec $0 \leq \beta \leq 1$:

$$TP(\beta) = CD + \beta ID \tag{3.5}$$

$$FP(\beta) = FD + (1-\beta)ID \tag{3.6}$$

$$FN(\beta) = MD + (1-\beta)ID \tag{3.7}$$

En remplaçant les équations 3.5, 3.6 et 3.7 dans les équations 3.3 on obtient :

$$P(\beta) = \frac{CD + \beta ID}{CD + \beta ID + FD + (1-\beta)ID} = \underbrace{\frac{CD}{CD + ID + FD}}_{C_p} + \beta \underbrace{\frac{ID}{CD + ID + FD}}_{I_p} \tag{3.8}$$

Avec C_p et I_p respectivement les parties consistantes et inconsistantes de la précision.

$$R(\beta) = \frac{CD + \beta ID}{CD + \beta ID + MD + (1-\beta)ID} = \underbrace{\frac{CD}{CD + ID + MD}}_{C_r} + \beta \underbrace{\frac{ID}{CD + ID + MD}}_{I_r} \tag{3.9}$$

Avec C_r et I_r respectivement les parties consistantes et inconsistantes du rappel.

Pour adapter ces mesures $R(\beta)$ et $P(\beta)$, au calcul de cartes de profondeurs relatives, Palou & Salembier (2014) calculent CD et ID par extension du principe de correspondance bipartite pour l'évaluation de la détection de contours (Martin *et al.*, 2004). $R(\beta)$ et $P(\beta)$ ainsi obtenu sont appelé consistance d'ordre local.
$R(\beta)$ et $P(\beta)$ peuvent etre graphiquement représentés sous forme de segment dans un repère de rappel/précision, comme l'illustre la figure 3.7. Les courbes jaunes, ont une valeur F constante.

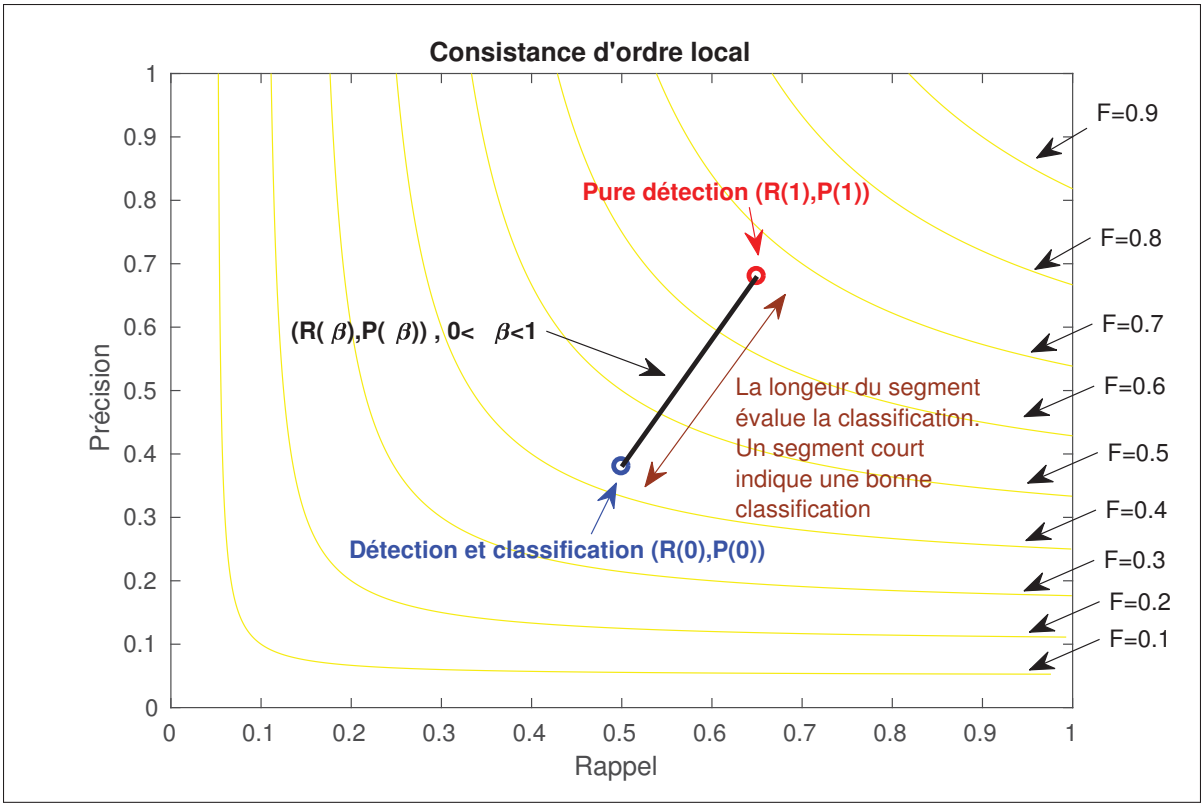


Figure 3.7 Illustration graphique de la consistance d'ordre local sur un plan rappel/précision

3.2.2.2 L'indice supérieur à l'aléatoire

Palou & Salembier (2014) définissent également un indice supérieur à l'aléatoire (*over random index*, ORI), qui compare les performances de l'assignation des ordres de profondeurs aux performances qu'aurait obtenues un classificateur aléatoire. Cet indice est défini comme suit :

$$ORI = max\left(0, \frac{F_{min} - F_{max}/2}{F_{max}/2}\right) \tag{3.10}$$

Avec $F_{max} = F(1, \theta_{max})$ et $F_{min} = F(0, \theta_{max})$, et $\theta_{max} = arg \max_{\theta} F(1, \theta)$ avec θ un paramètre du système évalué.

En effet, une classification aléatoire aurait, en théorie, 50% d'ordres de profondeurs correctes ($F_{min} = \frac{F_{max}}{2}$). Ainsi, un $ORI = 0$ indique que la méthode n'attribue pas plus d'ordre correct

qu'aurait potentiellement fait une attribution aléatoire, alors qu'un $ORI = 1$ indique que les ordres attribués sont tous corrects.

3.2.3 Résultat de l'évaluation de l'ordre

La méthode proposée a été testée sur les deux bases de données CMU et BVSD. Le paramètre N_{max}, qui fixe le nombre maximum de régions de la carte de profondeur résultante, ainsi que le paramètre S, représentant la taille moyenne en pixels de l'aire des superpixels de départ, ont été variés afin de voir leur impact. Les mouvements ont été calculés par la version modifiée de l'EpicFlow avec l'interpolation NW.

3.2.3.1 Base de données de Carnegie Mellon

Pour cette base de données CMU, en conséquence à la qualité de ses images, l'image précédente (I_{t-1}) et l'image suivante (I_{t+1}) ont été respectivement remplacées par la première et dernière trame de la séquence (La même démarche a été suivie par (Palou, 2013)).
Le paramètre N_{max} a été varié de 10 à 60 par pas de 10 et le paramètre S de 5 à 25 par pas de 5.

3.2.3.1.1 Résultats globaux

En appliquant la méthode proposée sur la base de données CMU, nous obtenons un ORI de 0.509. La figure 3.8 indique la consistance d'ordre locale obtenue.
Des cartes de profondeur relatives ont également été générées à partir de la vérité terrain de la segmentation. Ces résultats sont représentés par le segment noir sur la figure 3.8. Pour cette configuration, les étapes de la partition initiale, de la construction de l'arbre BPT et celle de l'élagage ne sont pas effectuée. En effet, ce sont les régions définies par la segmentation vérité terrain qui sont ordonnées à partir des pixels cachés/cachants. Ainsi, le résultat de cette configuration ne dépendent pas des parametres N_{max} et S. L'ORI de cette configuration est 0.539.

Pour cette base de données, la méthode de Palou (2013) ne présente que des résultats qualitatifs. Ainsi, pour se comparer au mieux à cette méthode, une partie des résultats qualitatifs de la méthode Palou (2013) sont présentés à la section suivante à la figure 3.9.

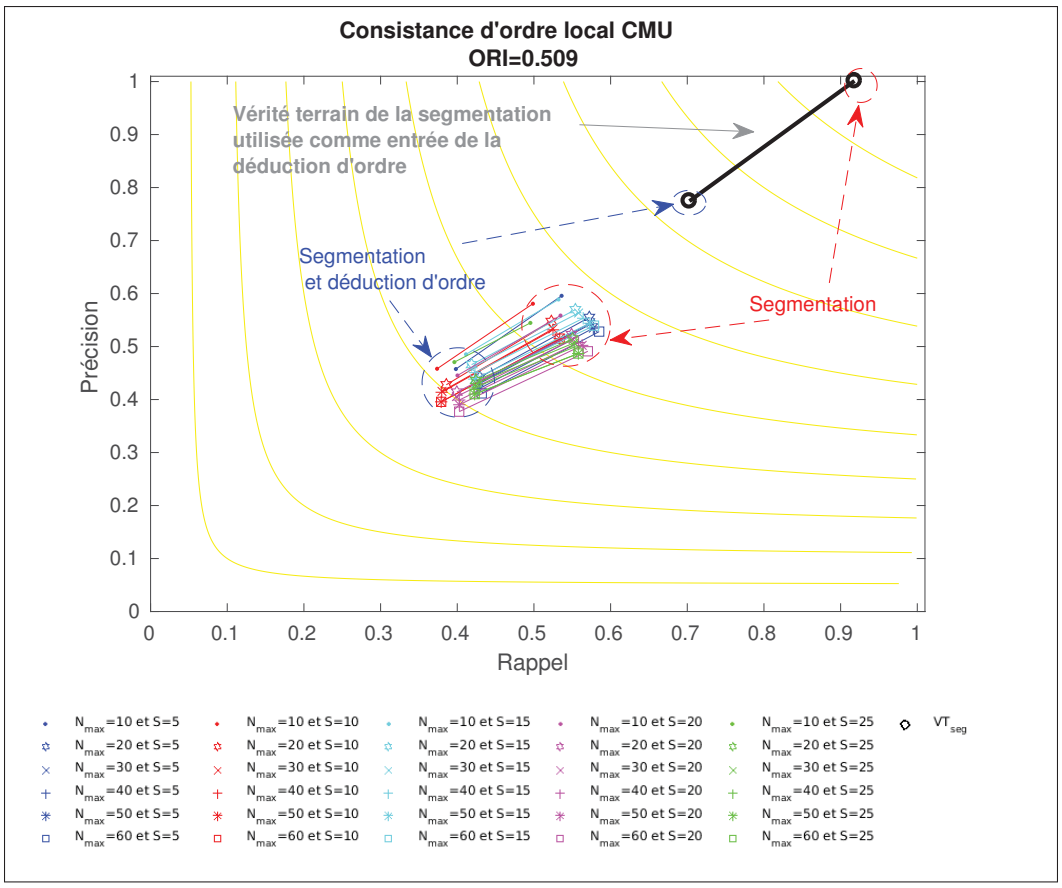


Figure 3.8 Consistance locale des séquences de la base de données CMU

3.2.3.1.2 Résultats détaillés pour quelques séquences

Les figures 3.9 et 3.10 représentent les résultats obtenus pour 8 séquences de la base de données CMU. Ces figures contiennent, pour chaque séquence : la trame traitée, la vérité terrain de la segmentation et de la carte de profondeur, la carte obtenue en considérant la vérité terrain de la segmentation, la carte de profondeur obtenue par la méthode proposée ainsi que la courbe de

consistance d'ordre local.

La légende des courbes de consistance d'ordre local est la même que celle de la figure 3.8.

La figure 3.9, contient également les cartes d'ordre obtenues par la méthode Palou (2013) pour ces 4 séquences.

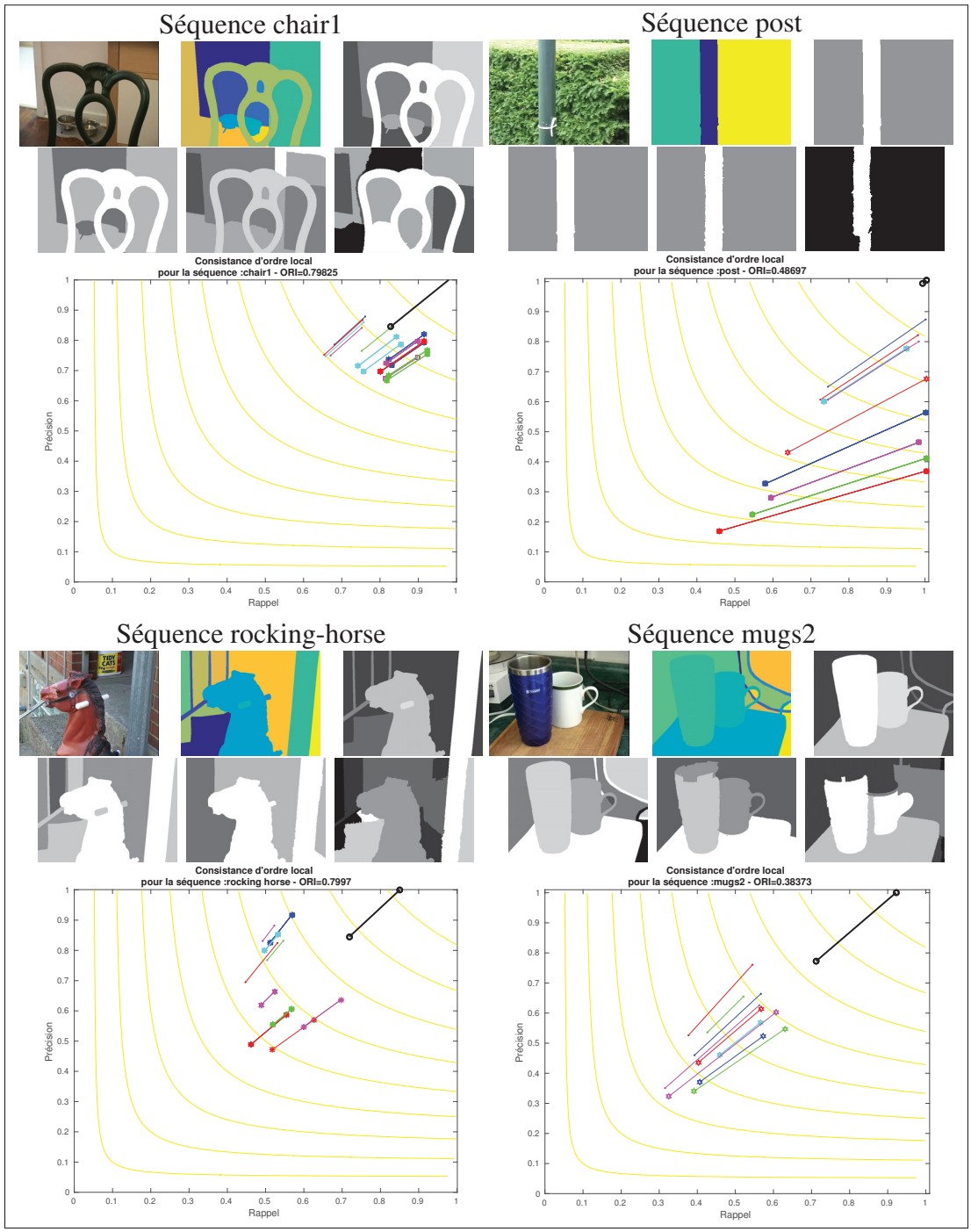


Figure 3.9 Consistance d'ordre local pour les séquences *chair*1, *rocking – horse*, *post* et *hand*2 de la base de données CMU

Pour chaque séquence, de gauche à droite, de haut en bas est représenté : la séquence traitée, les vérités terrain de la segmentation et de l'ordre, l'ordre obtenu avec la vérité terrain de segmentation, l'ordre obtenu par la méthode proposée (celui de l'ORI) ainsi que l'ordre obtenu par Palou (2013)

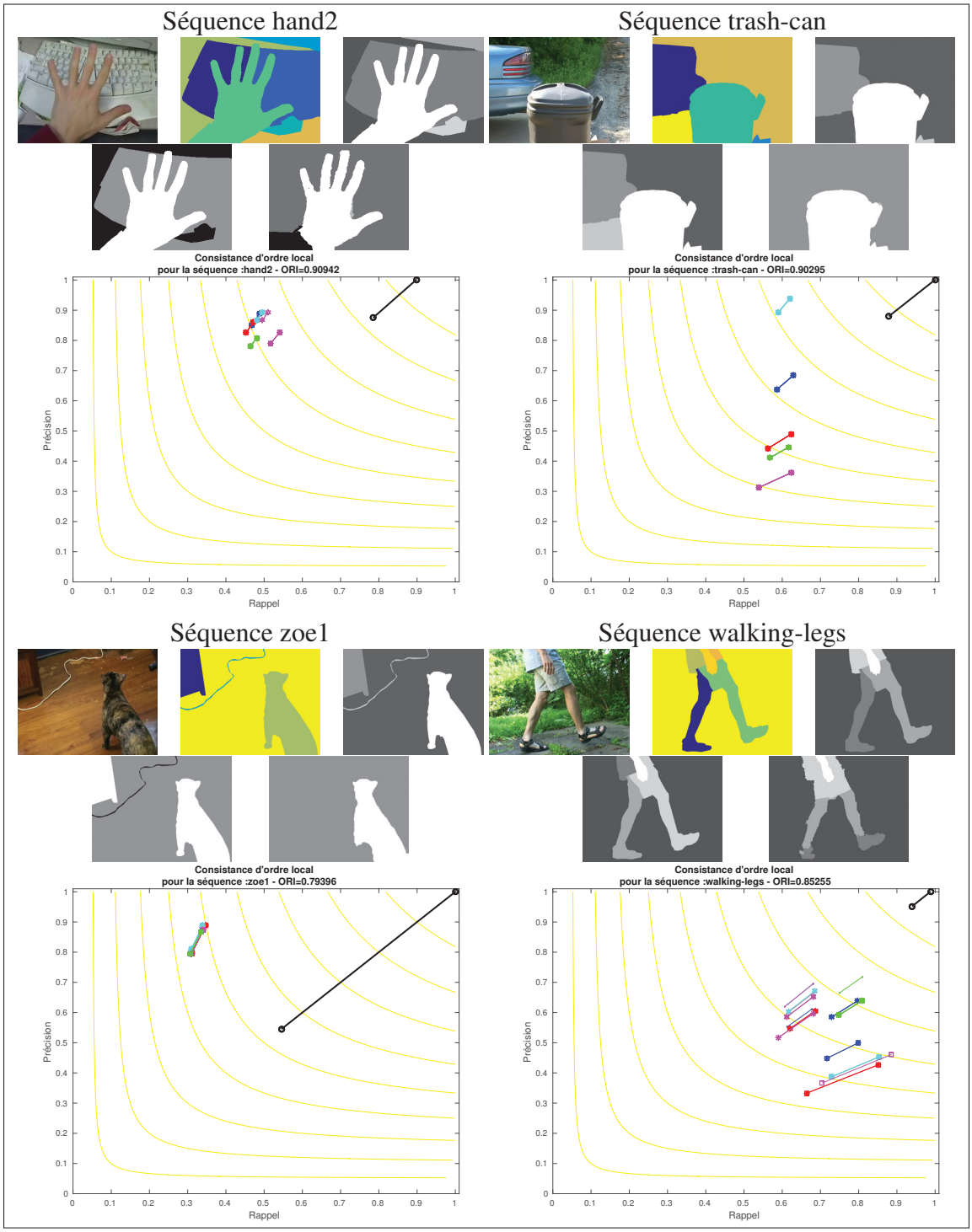


Figure 3.10 Consistance d'ordre local pour les séquences *mugs*2, *zoe*1, *trash* − *can* et *walking* − *legs* de la CMU
Pour chaque séquence, de gauche à droite, de haut en bas est représenté : la séquence traitée, les vérités terrain de la segmentation et de l'ordre, l'ordre obtenu avec la vérité terrain de segmentation ainsi que l'ordre (celui de l'ORI)

3.2.3.2 Base de données de Berkeley pour la segmentation vidéo

Pour cette base de données. Le paramètre N_{max} à été varié de 20 à 60 par pas de 20 et le paramètre S de 10 à 30 par pas de 10.

3.2.3.2.1 Résultats globaux

En appliquant la méthode proposée sur la base de données BVSD, nous obtenons un ORI moyen de 0.27, alors que la méthode de Palou (2013) obtient 0.25, on peut donc remarquer une petite amélioration. La figure 3.11 indique la consistance d'ordre local obtenue.

3.2.3.2.2 Résultats détaillées pour quelques séquences

Les figures 3.12 et 3.13 représentent les résultats obtenus pour 8 séquences de la base de données BVSD.

Ces figures contiennent, pour chaque séquence : la trame traitée, la vérité terrain de la segmentation et de la carte de profondeur, la carte obtenue en considérant la vérité terrain de la segmentation, la carte de profondeur obtenue par la méthode proposée ainsi que la courbe de consistance d'ordre local. La légende des courbes de consistance d'ordre local est la même que celle de la figure 3.11.

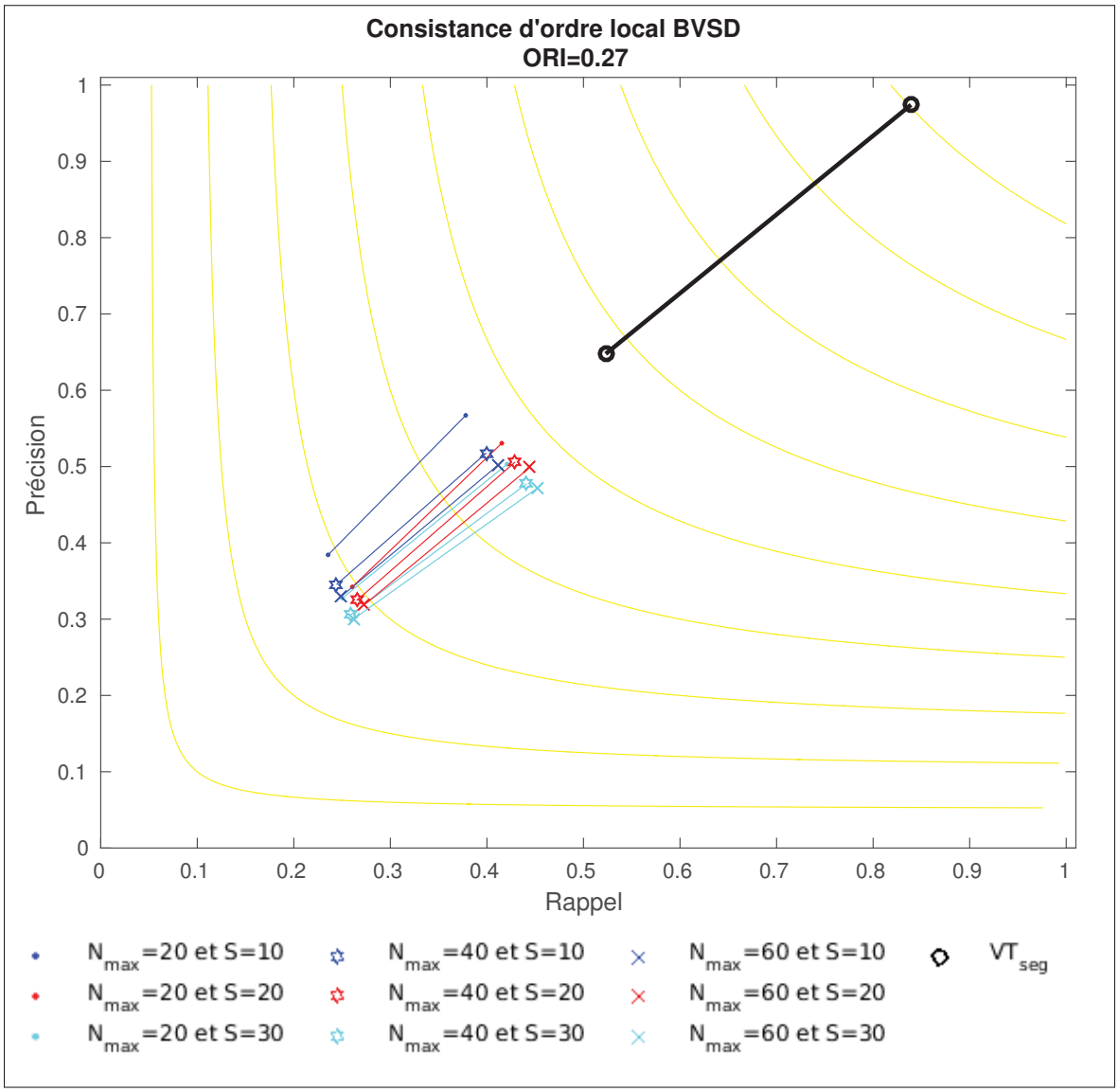


Figure 3.11 Consistance d'ordre locale des séquences de la base de données BVSD

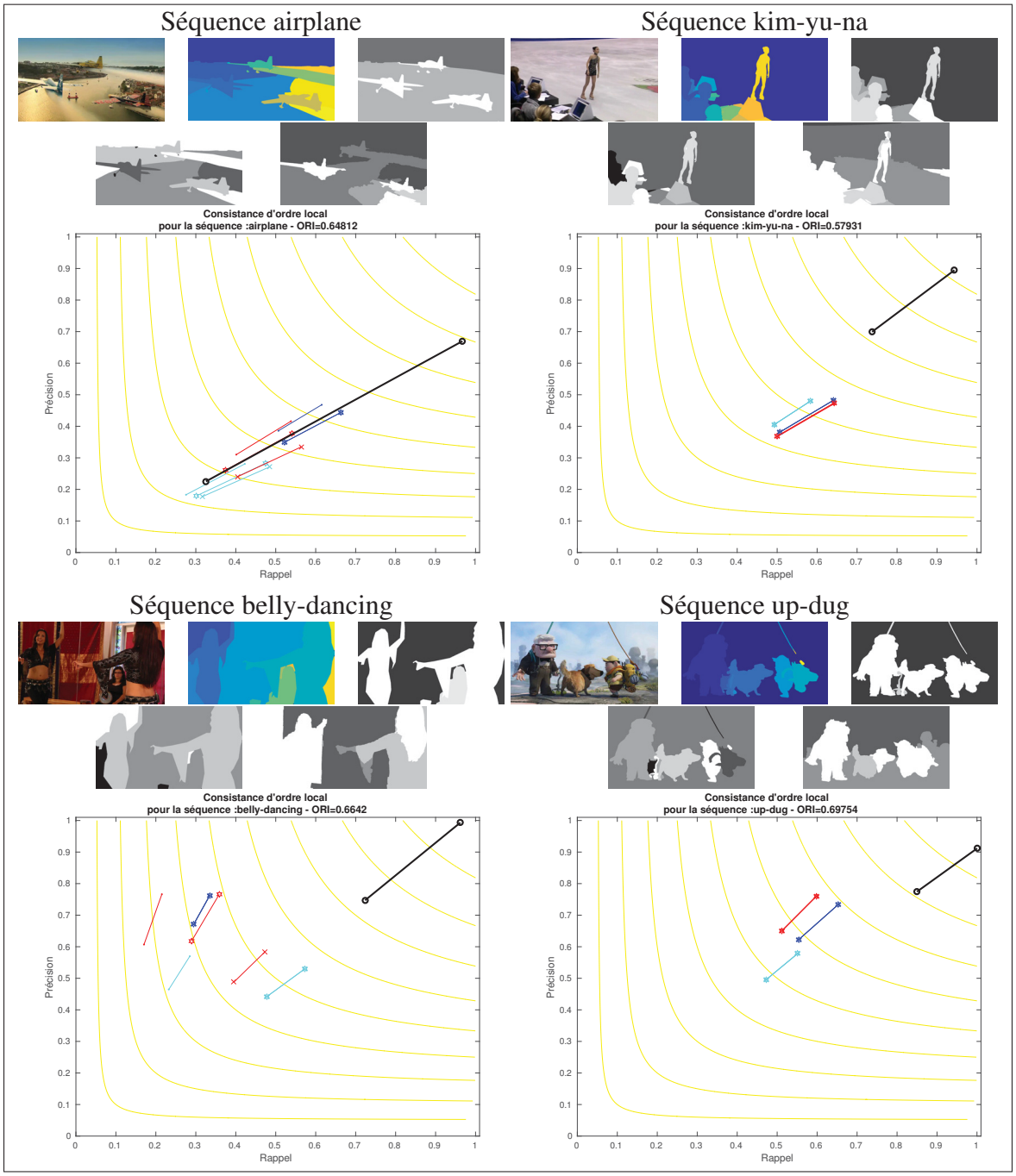


Figure 3.12 Consistance d'ordre local pour les séquences *airplane*, *kim – yu – na*, *belly – dancing* et *up – dug* de la BVSD
Pour chaque séquence, de gauche à droite, de haut en bas est représenté : la séquence traitée, les vérités terrain de la segmentation et de l'ordre, l'ordre obtenu avec la vérité terrain de segmentation ainsi que l'ordre (celui de l'ORI)

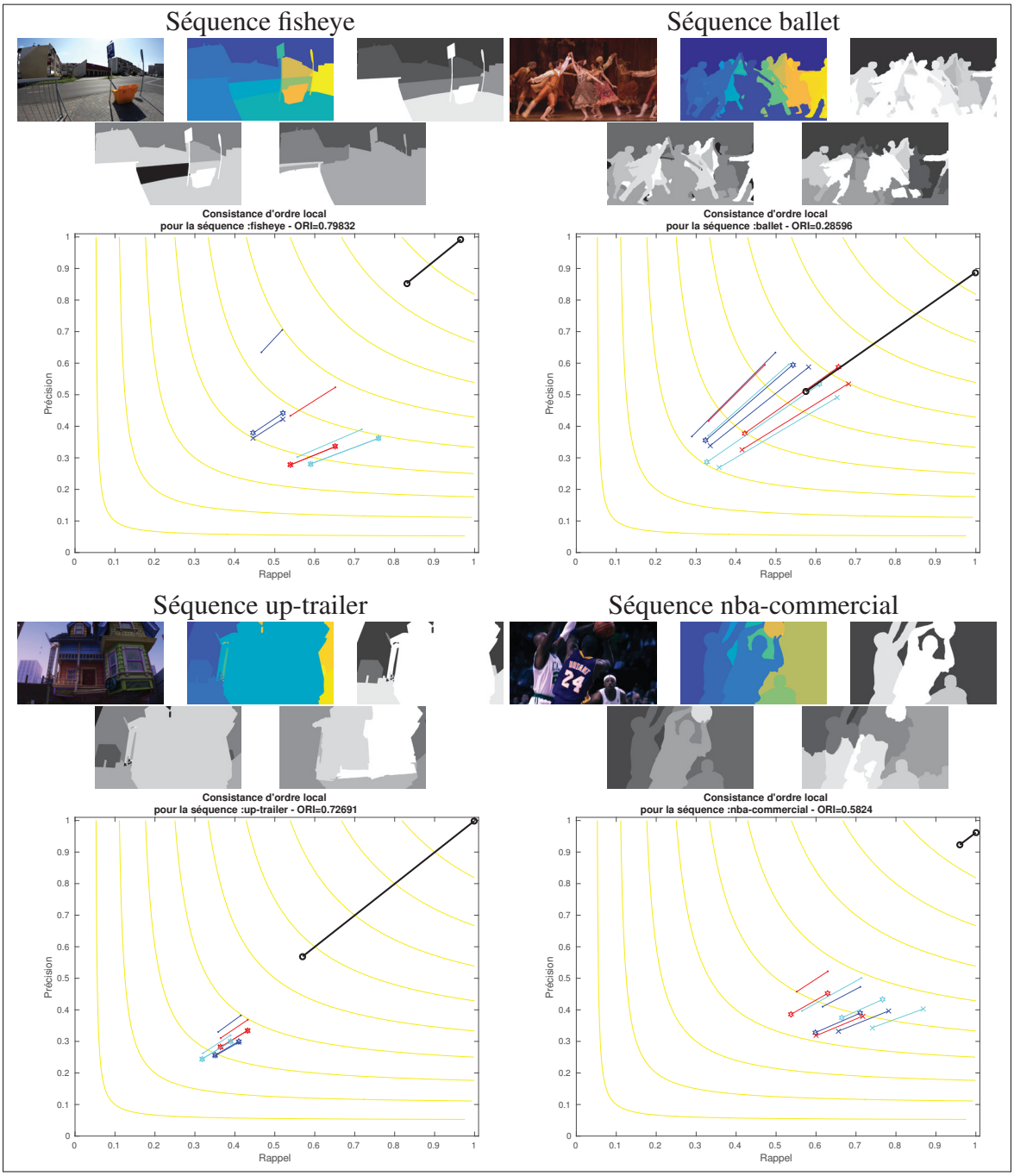


Figure 3.13 Consistance d'ordre local pour les séquences *fisheye*, *up − trailer*, *ballet* et *nba − commercial* de la BVSD
Pour chaque séquence, de gauche à droite, de haut en bas est représenté : la séquence traitée, les vérités terrain de la segmentation et de l'ordre, l'ordre obtenu avec la vérité terrain de segmentation ainsi que l'ordre (celui de l'ORI)

3.2.4 Discussion des résultats de la déduction d'ordre

Il peut sembler surprenant, que pour certaines séquences, l'ORI obtenu par la méthode au complet soit supérieur à l'ORI obtenu avec la vérité terrain de la segmentation. Comme c'est la cas dans les séquences *zoe*1 de la base donnée CMU, ou *airplane* de la base de donnée *BVSD* ; En effet pour ces séquences, le segment noir représentant les résultats obtenus en utilisant la vérité terrain de la segmentation est beaucoup plus long que les autres segments de la figure. Il y a deux explications à ce fait, la première est que la méthode au complet ne garde que des régions qui peuvent être ordonnées entre elles (grâce à l'étape de l'élagage). Ainsi à la séquence *zoe*1, la région correspondant au câble blanc ne se retrouve pas dans la carte de profondeur obtenue par la méthode au complet, mais elle est incorrectement ordonnée à la carte issue de la segmentation vérité terrain. La seconde explication est que les contours de la segmentation vérité terrain peuvent être légèrement décalés de ceux détectés par la méthode et ainsi perturber la déduction de l'ordre.

On pourrait également s'attendre à ce que les cartes de profondeur obtenues en utilisant la vérité terrain de la segmentation comme point départ aient des évaluations de segmentation parfaite. Pourtant, le haut du segment noir sur les figures n'est pas toujours au point $(R = 1, P = 1)$. La raison de ce résultat est que cette évaluation compare les régions définies par une différence de profondeur au niveau de la carte de profondeur obtenues à ceux présentes sur la vérité terrain de la carte de profondeur. Ainsi, si deux régions mitoyennes au niveau de la vérité terrain de segmentation leurs sont attribué le même ordre de profondeur, une frontière de segmentation est perdue (Ce qui aura pour effet $R < 1$). L'obtention de $P < 1$ se produit quand deux régions mitoyennes de la vérité terrain de la segmentation ont le même ordre de profondeur sur la vérité terrain de la carte de profondeur mais un ordre différent sur la carte de profondeur obtenue.

On remarque également, sans grande surprise, que pour la plupart des cas, la meilleure segmentation est obtenue pour la plus petite taille S de superpixels utilisée.

Quant au nombre maximum d'objets de la segmentation N_{max}, on peut remarquer qu'il n'a aucune influence sur certaines séquences (comme *zoe*1, $trash-can$, *hand*2 de la base de données

CMU, et $kim-yu-na$, $up-dug$ de la base de données BVSD). La raison est que pour ces séquences il n'y a du mouvement, et par conséquence des occlusions, qu'au niveau des objets en avant-plan. Sur d'autres séquences (comme $walking-legs$, $rocking-horse$ de la base de données CMU, et $fisheye$, $ballet$ de la base de données BVSD), l'effet du paramètre N_{max} est pour la plupart des cas, une augmentation du rappel et une diminution de la précision quand N_{max} est augmenté, vu que la segmentation comporte dans ce cas un plus grand nombre de régions.

On peut également remarquer que le mouvement présent à la trame traitée a une grande influence sur les performances obtenues. À titre d'exemple, la méthode obtient de mauvais résultats sur la séquence $birds-of-paradise$ de la base de données BVSD, car il n'y a presque pas de mouvement entre la trame traitée et celles qui la précède et succède. Par contre, en remplaçant la trame précédente et la trame suivante par celles prises à un intervalle de 10 trames, le résultat obtenu est plus concluant, comme l'illustre la figure 3.14.

Il est aussi à noter que la segmentation obtenue par l'approche suivie n'est pas une segmentation d'objets, mais de plans de profondeur. Même deux régions voisines distinctes en couleurs sont fusionnées si des occlusions ne sont pas détectées à leur frontière commune. Comme on peut le remarquer, avec les barreaux derrière le cheval en bois sur la séquence $rocking-horse$ de la base de données CMU (voir figure 3.9), ou les câbles en arrière plan de la séquence $up-dug$ de la base de données BVSD (voir figure 3.12).

L'approche proposée est pensée pour des scènes constituées de plans de profondeur distincts, où des objets seraient l'un en avant de l'autre. Ainsi, il est parfois compliqué d'attribuer une profondeur quand la séquence traitée ne respecte pas ce modèle. On peut citer l'exemple de la séquence *mugs*2 de la base de données *CMU* (présentée à la figure 3.10). En effet, cette séquence comporte deux tasses posées sur une table. La prise de vue de la séquence fait en sorte qu'en bas de l'image c'est la table qui est en avant, alors qu'un peu plus haut, les tasses sont sur la table et ce sont elles qu'on peut considérer en avant. Dans ces configurations, le résultat obtenu par la méthode proposée est imprévisible.

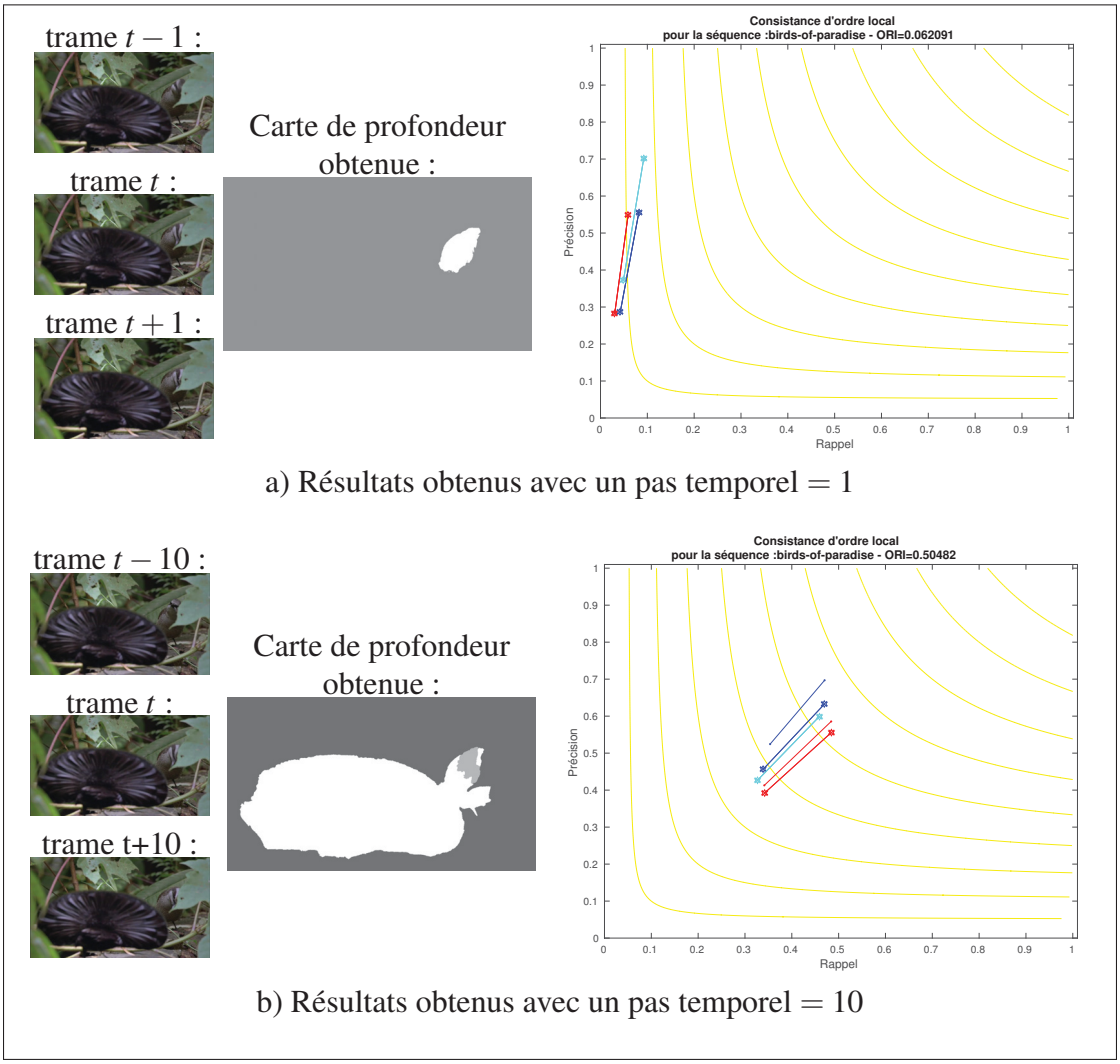


Figure 3.14 Illustration de l'effet de présence de mouvement

CONCLUSION ET RECOMMANDATIONS

Les travaux de cette recherche se sont penchés sur la conversion 2D à 3D par détection automatique des occlusions dynamiques. Après une étude de la littérature de ce contexte présentée au premier chapitre, le second chapitre a présenté la méthode proposée.

En effet, afin d'utiliser les occlusions dynamiques pour l'obtention d'un ordre relatif, il est essentiel d'effectuer la détection de ces occlusions. La méthode de détection des occlusions proposée nécessite un flot optique qui préserve les contours et qui est cohérent avant-arrière. Ce flot optique a été obtenu en modifiant le flot optique EpicFlow (Revaud *et al.*, 2015) qui a l'avantage de préserver les contours, et dont les modifications apportées ont permis de le rendre cohérent avant-arrière.

La génération de cartes de profondeur relative nécessite également une étape de segmentation pour définir les régions que les relations d'occlusions peuvent ordonner. Cette segmentation a été obtenue selon un schéma semblable à l'approche suivie par (Salembier & Palou, 2014), adaptée pour considérer les occlusions précédemment calculées. Cette adaptation a permis de supprimer l'estimation du flot optique par région de la méthode de (Salembier & Palou, 2014).

Le dernier chapitre a permis d'exposer les résultats qu'on obtient par la méthode proposée. L'évaluation de l'EpicFlow modifié sur la base de données de Middlebury a montré qu'il n'y a pas de pertes de performance par rapport à l'EpicFlow original (Revaud *et al.*, 2015). Cette modification de l'EpicFlow a également apporté la cohérence avant-arrière au flot optique. Quant à l'évaluation de l'ordre, effectuée sur les deux bases de données CMU et BVSD, elle a pu montrer que l'ordre pouvait être obtenu en se basant sur la méthode de détection des occlusions proposée alors que celle-ci ne nécessite pas d'estimation de flot optique par région.

Une limitation de l'approche proposée est que la cohérence dans le temps n'est pas assurée. En effet, le traitement de deux trames consécutives peut donner des régions de segmentation

très différentes. Une des causes de cette incohérence est qu'un objet peut arrêter d'être en mouvement, ce qui a pour effet que des occlusions ne seront plus détectées à ses contours (pour une configuration en caméra fixe). Il résultera de cette absence d'occlusions que la région de cet objet sera fusionnée avec d'autres régions et par conséquent la carte de profondeur générée sera différente de celle de la trame précédente. L'approche suivie propose d'effectuer le traitement sur chaque trame d'une séquence. Il serait possible d'envisager une propagation de profondeur aux trames voisines, selon une des approches de conversion semi-automatique de conversion 2D à 3D (présentées à la sous-section 1.2.1). Ce qui permettrait d'atténuer le manque de cohérence dans le temps, en plus de réduire les coûts de calculs.

Cette approche a exploité les occlusions dynamiques pour générer des cartes de profondeur relative. D'autres indices pourraient y être fusionnés pour améliorer la détection de l'ordre. Bien que moins fiables, les occlusions statiques pourraient être utilisées en absences d'occlusions dynamiques. Une autre possibilité serait de combiner l'ordre obtenu avec un indice de profondeur permettant d'avoir des indications sur la profondeur absolue. En effet, il serait envisageable d'appliquer un algorithme de SFM à chaque région obtenue de la carte de profondeur relative. Le flot optique servirait à effectuer une compensation de mouvement afin de respecter la contrainte des algorithmes de SFM adaptés à des scènes fixes.

BIBLIOGRAPHIE

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P. & Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2274-2282. doi : 10.1109/TPAMI.2012.120.

Alonso-González, A. (2014). *Multidimensional SAR data analysis based on binary partition trees and the covariance matrix geometry*. (Thèse de doctorat). Repéré à http://upcommons.upc.edu/handle/2117/95357.

Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J. & Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1), 1–31. doi : 10.1007/s11263-010-0390-2.

Battiato, S., Curti, S. & Cascia, M. L. (2004). *Depth map generation by image classification*. Proceedings of SPIE Three-Dimensional Image Capture and Applications VI. doi : 10.1117/12.526634.

Bay, H., Tuytelaars, T. & Van Gool, L. (2006). SURF : Speeded up robust features. Dans Leonardis, A., Bischof, H. & Pinz, A. (Éds.), *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (vol. 3951 LNCS, pp. 404–417). Berlin, Heidelberg : Springer Berlin Heidelberg. doi : 10.1007/11744023_32.

Beucher, S. & Meyer, F. (1993). The morphological approach to segmentation : the watershed transformation. Mathematical morphology in image processing. *Optical Engineering*, 34, 433–481. doi : Export Date 6 May 2013.

Boykov, Y., Veksler, O. & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11), 1222–1239. doi : 10.1109/34.969114.

Brox, T. & Malik, J. (2011). Large displacement optical flow : Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3), 500–513. doi : 10.1109/TPAMI.2010.143.

Cao, X., Li, Z. & Dai, Q. (2011). Semi-automatic 2D-to-3D conversion using disparity propagation. *IEEE Transactions on Broadcasting*, 57(2 PART 2), 491–499. doi : 10.1109/TBC.2011.2127650.

Chen, J.-C. & Huang, M.-y. (2016). 2D-to-3D Conversion System using Depth Map Enhancement. *KSII Transactions on Internet & Information Systems*, 10(3), 1159–1181. doi : 10.3837/tiis.2016.03.012.

Chen, J., Zhao, J., Wang, X., Huang, C., Dong, E., Chen, B. & Yuan, Z. (2011). A simple semi-automatic technique for 2D to 3D video conversion. *Lecture Notes in Computer*

Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7003 LNAI(PART 2), 336–343. doi : 10.1007/978-3-642-23887-1_42.

Cheng, F.-h. & Sung, T.-y. (2013). Merging Static and Dynamic Depth Cues with Optical-Flow. *Mathematical Problems in Engineering*, 2013, 12.

Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, I(3), 886–893. doi : 10.1109/CVPR.2005.177.

Dekel (Basha), T., Moses, Y. & Avidan, S. (2014). Photo Sequencing. Dans Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y. & Schmid, C. (Éds.), *International Journal of Computer Vision* (vol. 110, pp. 275–289). Berlin, Heidelberg : Springer Berlin Heidelberg. doi : 10.1007/s11263-014-0712-x.

Dimiccoli, M. & Salembier, P. (2009). Exploiting T-junctions for depth segregation in single images. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 1229–1232. doi : 10.1109/ICASSP.2009.4959812.

Dollár, P. & Zitnick, C. L. (2015). Fast Edge Detection Using Structured Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8), 1558-1570. doi : 10.1109/TPAMI.2014.2377715.

Fehn, C. (2004). Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. *Stereoscopic Displays and Virtual Reality Systems XI*, 5291, 93–104. doi : 10.1117/12.524762.

Feng, Y., Ren, J. & Jiang, J. (2011). Object-based 2D-to-3D video conversion for effective stereoscopic content generation in 3D-TV applications. *IEEE Transactions on Broadcasting*, 57(2 PART 2), 500–509. doi : 10.1109/TBC.2011.2131030.

Fu, H., Wang, C., Tao, D. & Black, M. J. (2016). Occlusion Boundary Detection via Deep Exploration of Context. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 241–250. doi : 10.1109/CVPR.2016.33.

Gautama, T. & Van Hulle, M. M. (2002). A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Transactions on Neural Networks*, 13(5), 1127–1136. doi : 10.1109/TNN.2002.1031944.

Guttmann, M., Wolf, L. & Cohen-Or, D. (2009). Semi-automatic stereo extraction from video footage. *Proceedings of the IEEE International Conference on Computer Vision*, (Iccv), 136–142. doi : 10.1109/ICCV.2009.5459158.

Hartley, R. & Zisserman, A. (2003). *Multiple View Geometry in Computer Vision* (éd. Second). Cambridge University Press, ISBN : 0521540518.

Herrera, J. L., Del-Blanco, C. R. & García, N. (2016). A novel 2D to 3D video conversion system based on a machine learning approach. *IEEE Transactions on Consumer Electronics*, 62(4), 429–436. doi : 10.1109/TCE.2016.7838096.

Horn, B. K. P. & Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17(1), 185–203. doi : https://doi.org/10.1016/0004-3702(81)90024-2.

Huang, W., Cao, X., Lu, K., Dai, Q. & Bovik, A. C. (2015). Toward naturalistic 2D-to-3D conversion. *IEEE Transactions on Image Processing*, 24(2), 724–733. doi : 10.1109/TIP.2014.2385474.

Huang, X., Wang, L., Huang, J., Li, D. & Zhang, M. (2009). A depth extraction method based on motion and geometry for 2D to 3D conversion. *3rd International Symposium on Intelligent Information Technology Application, IITA 2009*, 3, 294–298. doi : 10.1109/IITA.2009.481.

Jebara, T., Azarbayejani, A. & Pentland, A. (1999). 3D structure from 2D motion. *IEEE Signal Processing Magazine*, 16(3), 66-84. doi : 10.1109/79.768574.

Jia, Z., Gallagher, A., Chang, Y. J. & Chen, T. (2012). A learning-based framework for depth ordering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 294–301. doi : 10.1109/CVPR.2012.6247688.

Ju, K. & Xiong, H. (2014). A semi-automatic 2D-to-3D video conversion with adaptive key-frame selection. 9273, 92730M. doi : 10.1117/12.2071947.

Ju, K., Wang, B. & Xiong, H. (2016). Structure-aware priority belief propagation for depth estimation. *2015 Visual Communications and Image Processing, VCIP 2015*. doi : 10.1109/VCIP.2015.7457889.

Jung, C., Wang, L., Zhu, X. & Jiao, L. (2015). 2D to 3D conversion with motion-type adaptive depth estimation. *Multimedia Systems*, 21(5), 451–464. doi : 10.1007/s00530-014-0375-z.

Karsch, K., Liu, C. & Bing Kang, S. (2015). Depth transfer : Depth extraction from videos using nonparametric sampling. *Dense Image Correspondences for Computer Vision*, 36(11), 173–205. doi : 10.1007/978-3-319-23048-1_9.

Kim, J., Baik, A., Jung, Y. J. & Park, D. (2010). 2D-to-3D conversion by using visual attention analysis. *Displays*, 7524, 752412. doi : 10.1117/12.839925.

Knorr, S., Imre, E., Zkalayci, B. Ö., Alatan, A. A. & Sikora, T. (2007). A modular scheme for 2D/3D conversion of TV broadcast. *Proceedings - Third International Symposium on 3D Data Processing, Visualization, and Transmission, 3DPVT 2006*, 703–710. doi : 10.1109/3DPVT.2006.15.

Koenderink, J. J., Van Doorn, A. J., Kappers, A. M. & Todd, J. T. (2001). Ambiguity and the 'mental eye' in pictorial relief. *Perception*, 30(4), 431–448. doi : 10.1068/p3030.

Konrad, J., Wang, M., Ishwar, P., Wu, C. & Mukherjee, D. (2013). Learning-based, automatic 2D-to-3D image and video conversion. *IEEE Transactions on Image Processing*, 22(9), 3485–3496. doi : 10.1109/TIP.2013.2270375.

Kowdle, A., Gallagher, A. & Chen, T. (2013). Revisiting depth layers from occlusions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2091–2098. doi : 10.1109/CVPR.2013.272.

Lai, Y. K., Lai, Y. F. & Chen, Y. C. (2013). An effective hybrid depth-generation algorithm for 2D-to-3D conversion in 3D displays. *IEEE/OSA Journal of Display Technology*, 9(3), 154–161. doi : 10.1109/JDT.2012.2224637.

Levina, E. & Bickel, P. (2001). The Earth Mover's distance is the Mallows distance : Some insights from statistics. *Proceedings of the IEEE International Conference on Computer Vision*, 2, 251–256. doi : 10.1109/ICCV.2001.937632.

Li, Z., Cao, X. & Dai, Q. (2012). A novel method for 2D-to-3D video conversion using bi-directional motion estimation. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 1429–1432. doi : 10.1109/ICASSP.2012.6288159.

Liao, M., Gao, J., Yang, R. & Gong, M. (2012). Video stereolization : Combining motion analysis with user interaction. *IEEE Transactions on Visualization and Computer Graphics*, 18(7), 1079–1088. doi : 10.1109/TVCG.2011.114.

Liu, C., Yuen, J. & Torralba, A. (2015a). Sift flow : Dense correspondence across scenes and its applications. *Dense Image Correspondences for Computer Vision*, 33(5), 15–49. doi : 10.1007/978-3-319-23048-1_2.

Liu, W., Wu, Y., Guo, F. & Hu, Z. (2015b). An Efficient Approach for 2D to 3D Video Conversion Based on Structure from Motion. *Vis. Comput.*, 31(1), 55–68. doi : 10.1007/s00371-013-0904-3.

Liu, X., Mao, X., Yang, X., Zhang, L. & Wong, T.-T. (2013). Stereoscopizing cel animations. *ACM Transactions on Graphics*, 32(6), 1–10. doi : 10.1145/2508363.2508396.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91–110. doi : 10.1023/B:VISI.0000029664.99615.94.

Martin, D. R., Fowlkes, C. C. & Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5), 530–549. doi : 10.1109/TPAMI.2004.1273918.

Ming, A., Wu, T., Ma, J., Sun, F. & Zhou, Y. (2016). Monocular Depth-Ordering Reasoning with Occlusion Edge Detection and Couple Layers Inference. *IEEE Intelligent Systems*, 31(2), 54–65. doi : 10.1109/MIS.2015.94.

Ogale, A. S. & Aloimonos, Y. (2005). Shape and the stereo correspondence problem. *International Journal of Computer Vision*, 65(3), 147–162. doi : 10.1007/s11263-005-3672-3.

Oliva, A. & Torralba, A. (2001). Modeling the shape of the scene : A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3), 145–175. doi : 10.1023/A:1011139631724.

Palou, G. (2013). *Monocular Depth Estimation in Images and Sequences Using Occlusion Cues*. (Thèse de doctorat). Repéré à http://hdl.handle.net/10803/144653.

Palou, G. & Salembier, P. (2011). Occlusion-based depth ordering on monocular images with binary partition tree. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 1093–1096. doi : 10.1109/ICASSP.2011.5946598.

Palou, G. & Salembier, P. (2013). Monocular depth ordering using t-junctions and convexity occlusion cues. *IEEE Transactions on Image Processing*, 22(5), 1926–1939. doi : 10.1109/TIP.2013.2240002.

Palou, G. & Salembier, P. (2014). Precision-Recall-Classification Evaluation Framework : Application to Depth Estimation on Single Images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8689, 648–662. doi : 10.1007/978-3-319-10590-1_42.

Revaud, J., Weinzaepfel, P., Harchaoui, Z. & Schmid, C. (2015). EpicFlow : Edge-Preserving Interpolation of Correspondences for Optical Flow . *CVPR*, 1164–1172. doi : 10.1063/1.4905777.

Rezaeirowshan, B., Ballester, C. & Haro, G. (2015). Monocular Depth Ordering using Perceptual Occlusion Cues. *International Conference on Computer Vision and Applications, VISAPP 2016*.

Salembier, P. & Garrido, L. (2000). Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions on Image Processing*, 9(4), 561–576. doi : 10.1109/83.841934.

Salembier, P. & Palou, G. (2014). Depth order estimation for video frames using motion occlusions. *IET Computer Vision*, 8(2), 152–160. doi : 10.1049/iet-cvi.2012.0287.

Schodl, A. & Essa, I. (2001). Depth layers from occlusions. *2001 IEEE Conference on Computer Vision and Pattern Recognition*, 9–14. Repéré à http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=990534.

Stein, A. N. & Hebert, M. (2009). Occlusion boundaries from motion : Low-level detection and mid-level reasoning. *International Journal of Computer Vision*, 82(3), 325–357. doi : 10.1007/s11263-008-0203-z.

Sun, L., Zhao, Q., Xiang, J., Shi, J., Wang, L., Hu, S. & Su, S. (2009). Adsorption of NO and NH3 over CuO/γ-Al2O3 catalyst by DRIFTS. *Huagong Xuebao/CIESC Journal*, 60(2), 444–449. doi : 10.1007/s11771.

Sundberg, P., Brox, T., Maire, M., Arbeláez, P. & Malik, J. (2011). Occlusion boundary detection and figure/ground assignment from optical flow. 2233-2240. doi : 10.1109/CVPR.2011.5995364.

Szeliski, R. (2011). *Computer Vision : Algorithms and Applications*. doi : 10.1007/978-1-84882-935-0.

Tang, C., Hou, C. & Song, Z. (2013). Defocus map estimation from a single image via spectrum contrast. *Optics Letters*, 38(10), 1706. doi : 10.1364/OL.38.001706.

Varekamp, C. & Barenbrug, B. (2007). Improved Depth Propagation for 2D to 3D Video Conversion using Key-frames. *European Conference on Visual Media Production*, 1–7. doi : 10.1049/cp:20070061.

Wang, D., Liu, J., Sun, J., Liu, W. & Li, Y. (2012). A novel key-frame extraction method for semi-automatic 2D-to-3D video conversion. *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB*, (708059), 2–6. doi : 10.1109/BMSB.2012.6264235.

Wasserman, L. (2004). *All of Statistics : A Concise Course in Statistical Inference*. New York : Springer. doi : 10.1007/978-0-387-21736-9.

Wei, Q. (2005). *Converting 2D to 3D : A Survey*. The Netherlands : Delft University of Technology.

Weinzaepfel, P., Revaud, J., Harchaoui, Z. & Schmid, C. (2013). DeepFlow : Large displacement optical flow with deep matching. *Proceedings of the IEEE International Conference on Computer Vision*, 1385–1392. doi : 10.1109/ICCV.2013.175.

Xie, J., Girshick, R. B. & Farhadi, A. (2016). Deep3D : Fully Automatic 2D-to-3D Video Conversion with Deep Convolutional Neural Networks. *CoRR*, abs/1604.03650. Repéré à http://arxiv.org/abs/1604.03650.

Yan, X., Yang, Y., Er, G. & Dai, Q. (2011). Depth map generation for 2D-to-3D conversion by limited user inputs and depth propagation. *3DTV Conference : The True Vision - Capture, Transmission and Display of 3D Video, 3DTV-CON 2011 - Proceedings*, 3–6. doi : 10.1109/3DTV.2011.5877167.

Yin, S., Dong, H., Jiang, G., Liu, L. & Wei, S. (2015). A Novel 2D-to-3D video conversion method using time-coherent depth maps. *Sensors (Switzerland)*, 15(7), 15246–15264. doi : 10.3390/s150715246.

Zhang, G., Jia, J., Wong, T. T. & Bao, H. (2008). Recovering consistent video depth maps via bundle optimization. *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 1–8. doi : 10.1109/CVPR.2008.4587496.

Zhang, L., Vázquez, C. & Knorr, S. (2011). 3D-TV content creation : Automatic 2D-to-3D video conversion. *IEEE Transactions on Broadcasting*, 57(2 PART 2), 372–383. doi : 10.1109/TBC.2011.2122930.

Zhang, Z., Yin, S., Liu, L. & Wei, S. (2015). A real-time time-consistent 2D-to-3D video conversion system using color histogram. *IEEE Transactions on Consumer Electronics*, 61(4), 524–530. doi : 10.1109/TCE.2015.7389808.