Stéréovision
Comment reconstituer le relief d'une scène ?

Flavien Léger
fleger@ens-cachan.fr

Thomas Pumir de Louvigny
tpumir@ens-cachan.fr

Pauline Tan
ptan@ens-cachan.fr

Encadrants

Jean-Michel Morel
CMLA
morel@cmla.ens-cachan.fr

Neus Sabater
CMLA
sabater@cmla.ens-cachan.fr

– 24 juin 2010 –

Remerciements

Nous remercions très chaleureusement notre maître de stage Jean-Michel Morel, pour nous avoir fait découvrir la stéréovision et l'univers de l'*image processing*, et plus généralement pour nous avoir ouvert la porte sur le travail de chercheur, parfois amusant, parfois frustrant, mais toujours riche en enseignement et plein de surprises.
Nous remercions également Neus Sabater, qui nous a guidé avec beaucoup de patience et d'indulgence durant tout ce stage, toujours disponible pour nos questions lors de notre séjour au CMLA.

Résumé

L'objectif en stéréoscopie est de calculer la profondeur d'une scène à partir de la donnée de deux images de cette scène. Il existe pour ce faire deux approches envisageables, mais nous nous concentrerons ici sur les méthodes dites *globales*, et plus particulièrement sur une méthode basée sur des coupes dans un graphe donné, qui a donné de bons résultats.

Table des matières

Introduction . . . 7
I Généralités . . . 7
I.1 Position du problème . . . 7
I.2 Disparité . . . 8
I.3 Géométrie épipolaire . . . 9
I.4 Phénomène d'occlusion . . . 10
I.5 Mise en correspondance . . . 11
II Méthodes globales, méthodes locales . . . 11
II.1 Méthodes globales . . . 11
II.2 Méthodes locales . . . 12
II.3 Stratégies complémentaires . . . 13
II.4 Le banc d'essai Middlebury . . . 14
III Problème du flot maximal . . . 15
III.1 Graphes et réseau . . . 15
III.2 Flot et coupe dans un graphe . . . 15
III.3 Théorème de Max-Flow/Min-Cut . . . 16
III.4 Algorithme de Ford-Fulkerson (variante) . . . 17
IV Méthode de Kolmogorov-Zabih basée sur le Graph Cut . . . 20
IV.1 Notations . . . 20
IV.2 Fonctionnelle d'énergie . . . 21
IV.3 Énergie et graphes . . . 23
IV.4 *α-expansion move* . . . 26
IV.5 Construction du graphe . . . 27
IV.6 Choix des paramètres . . . 32
IV.7 Mesure de dissimilarité de Birchfield et Tomasi . . . 34
V Résultats expérimentaux . . . 39
V.1 Images Middlebury . . . 39
V.2 Autres images . . . 41
Conclusion . . . 44
Références bibliographiques . . . 46

Glossaire

Assignement Couple de pixels pouvant potentiellement correspondre. Un assignement est dit actif dans une certaine configuration si les deux pixels sont mis en correspondance dans cette configuration, inactif sinon.

Coupe (dans un graphe) Pour une bipartition des nœuds donnée séparant la source et le puits du graphe, ensemble des arcs partant de la classe de la source et aboutissant dans la classe du puits. Une coupe peut être également définie par cette bipartition.

Disparité Vecteur de décalage entre deux pixels mis en correspondance. En géométrie épipolaire, la disparité peut être assimilée à son unique composante non nul, et sa valeur absolue est inversement proportionnel à la profondeur du point associé.

Géométrie épipolaire Voir Images rectifiées.

Image de référence L'une des deux images de la paire pour laquelle on cherche la carte de disparité.

Images rectifiées Images pour lesquelles chaque ligne de l'image de gauche correspond à la même ligne dans l'image de droite. On dit alors qu'elles sont en géométrie épipolaire. Cela implique en particulier que les axes (simulés) des deux caméras sont parallèles et que les images sont dans le même plan.

Mise en correspondance Fait d'associer à un pixel de l'image de référence son homologue (supposé) dans l'autre image.

Occlusion Phénomène décrivant le fait que certains points de la scène ne sont visibles que par l'une des deux caméras. De tels points sont dits occlus.

Scène Portion de l'espace dont on considère les images et dont on veut reconstituer le relief. Elle est continue par morceaux.

Notations

$A(f)$	ensemble des assignements actifs dans la configuration f
$\mathcal{A}^\alpha$	ensemble des assignements de décalage α
$d(\langle p,q\rangle)$	décalage (ou encore disparité) de l'assignement $\langle p,q\rangle$, c'est-à-dire $q-p$ (vectoriellement parlant)
$d_f(p)$	disparité du pixel p dans la configuration f
f	configuration ; $f(a)$ prend la valeur 1 si l'assignement a est actif
$I(p)$	intensité d'un pixel (ou moyenne des valeurs sur les trois canaux si l'image est en couleur (RBG)), de valeur comprise entre 0 et 255
$\langle p,q\rangle$	assignement (paire non ordonnée) où par convention, p désigne le pixel de l'image de gauche et q celui de l'image de droite

Introduction

Un des enjeux des technologies modernes est de réussir à reconstituer précisément le relief (ou la profondeur) d'une scène. On peut citer la reconstitution du relief d'une région donnée (après une catastrophe naturelle, par exemple), ou encore la vision des robots, qui seraient alors capables de se déplacer dans un environnement inconnu.

Une piste envisageable est l'utilisation du laser, il permet, en mesurant le temps de retour du rayon, d'estimer la distance parcourue par celui-ci avant de toucher un obstacle. Cette approche est toutefois mauvaise dans lorsque l'absorption du rayon lumineux dépend du matériau. Une autre approche consiste à mesurer physiquement le relief, en utilisant un bras mécanique qui irait toucher les objets et dont on connaîtrait précisément la position dans l'espace. Mais elle est excessivement coûteuse et donc peu accessible.

Dans ces conditions, l'une des pistes les plus prometteuses dans ce domaine est la stéréo-scopie. Ainsi que notre vue binoculaire nous en donne l'intuition, le relief peut être reconstitué de manière sûre avec deux prises de vue décalées d'une même scène. En effet, outre certains autres processus, nous percevons le relief grâce au fait qu'un objet proche se déplace beaucoup lorsque nous basculons de la vision de l'œil gauche à celui de l'œil droit (on peut tenter l'expérience en fixant un stylo tenu à bout de nez avec chacun des deux yeux). C'est sur ce principe que repose la stéréoscopie. Mais si le lien entre ce déplacement (appelé *disparité*) et la profondeur de l'objet est aisé d'expliciter, toute la difficulté réside dans le fait de mesurer ce déplacement. En effet, notre cerveau est capable de repérer dans les deux images qu'il reçoit de deux yeux les points qui correspondent au *même objet physique*, chose qu'un ordinateur ne sait pas faire. Cette association, que l'on appelle *mise en correspondance*, fait donc l'objet des recherches en stéréovision. Les approches envisagées sont classiquement réparties en deux catégories : les méthodes globales et les méthodes locales.

C'est dans ce cadre que nous présenterons, dans ce mémoire, un algorithme de méthode globales proposé par Kolmogorov et Zabih dans [16, 17, 15], et dont nous avons procédé à une mise en ligne du code sur le site IPOL [2]. L'intérêt de cette méthode, basée sur le principe des *α-expansion moves*, réside dans l'utilisation des coupes dans un graphe, qui permet de minimiser rapidement et efficacement une certaine énergie que nous expliciterons.

I Généralités

I.1 Position du problème

On dispose de deux images (image gauche et image droite) de la même scène, prises avec un décalage. Le but de la stéréoscopie binoculaire est de déterminer précisément le relief de cette scène, en fournissant (automatiquement) la carte du relief.

(a) Image gauche (b) Image droite (c) Vérité-terrain

Figure 1 – Exemple avec l'image Tsukuba. Les pixels les plus clairs correspondent aux points les plus proches. (On appelle *vérité-terrain* la carte réelle du relief.)

Il s'agit de reproduire les conditions dans lesquelles la reconstitution du relief est telle que notre cerveau la fait : deux yeux, donc deux vues d'une même scène légèrement décalées, puis une mise en

correspondance de chacun des points d'une image avec ceux de l'autre image (dans notre cerveau, cette opération se traduit par la fusion, généralement réussie, des deux images, nous conférant alors une vue dite *cyclopéenne*[1]).

I.2 Disparité

Lorsque l'on alterne la vue gauche et la vue droite d'une même scène, on peut remarquer que les objets les plus proches se déplacent plus que ceux qui sont plus éloignés. Ce déplacement est appelé *disparité*, et plus sa valeur est grande, plus l'objet est proche. Dans le cas de la vision humaine (où les axes optiques de chacun des yeux sont rigoureusement parallèles et les centres optiques dans un plan perpendiculaire à ces axes), ce déplacement n'a lieu que dans une seule direction, et on confond la valeur de ce déplacement scalaire avec la disparité ; la disparité donne alors exactement la profondeur du point considéré, la *baseline* (la droite qui supporte les centres optiques des caméras) étant alors parallèle aux plans de profondeur z. Ce cadre est celui de la *géométrie épipolaire* (cf partie suivante).

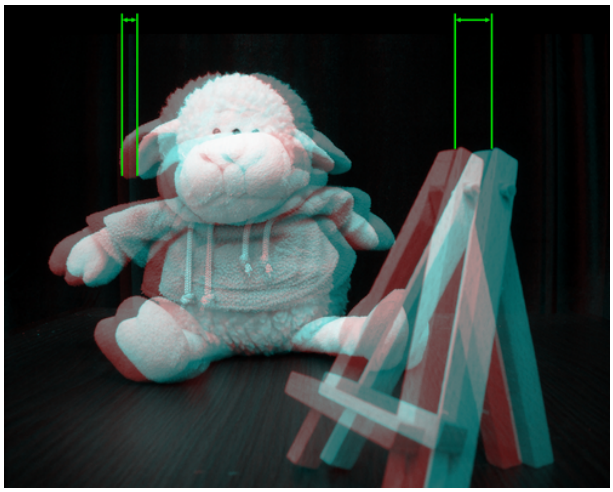
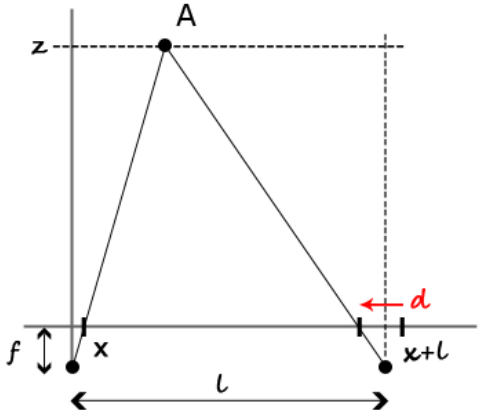


FIGURE 2 – En superposant les images gauche et droite (coloriées pour plus de lisibilité), on peut observer directement la disparité d'un point, qui correspond au *déplacement* de ce point (en vert). On remarque ici que l'oreille du mouton est plus éloigné que le chevalet, et présente une disparité plus petite.

Théorème I.1 *En géométrie épipolaire, si l'on connaît la distance entre les deux caméras, connaître la disparité, c'est connaître la profondeur.*

Preuve : Il suffit d'utiliser le théorème de Thalès pour montrer ce résultat.



1. Dans la mythologie grecque, les cylopes sont des géants qui ne possèdent qu'un seul œil au milieu du front.

Les deux caméras sont distantes de la longueur ℓ, de focale f ; connaître la disparité d nous donne alors la profondeur z du point A, par la relation : $\frac{z}{z+f} = \frac{\ell - d}{\ell}$, soit $\frac{d}{\ell} = \frac{f}{z+f}$. La disparité est donc inversement proportionnelle à la profondeur.

I.3 Géométrie épipolaire

Commençons par détailler l'acquisition des images d'une scène donnée par deux caméras. On modélise une caméra par un point, son *centre optique* et par un plan (qui correspond à son *plan image*) ; l'image d'un point de la scène par une caméra est donc la projection du point sur ce plan, suivant la direction de la droite qui passe par le centre optique.

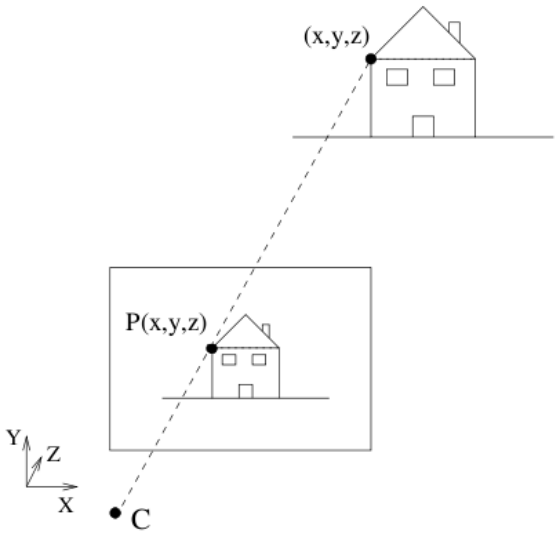


FIGURE 3 – Le point de l'espace (x, y, z) est projeté sur l'image en $P(x, y, z)$. Le centre optique de la caméra est ici C. Ce modèle est dit *modèle pin-hole*.

Considérons un point de l'espace X. Son image sur l'image de gauche (resp. de droite) est sa projection X_L (resp. X_R) sur le plan image de la caméra gauche (resp. droite). Inversement, connaissant les deux projetés, on connaît exactement X comme étant l'intersection des droites $(O_L\, X_L)$ et $(O_R\, X_R)$ (cf. Fig. 4).

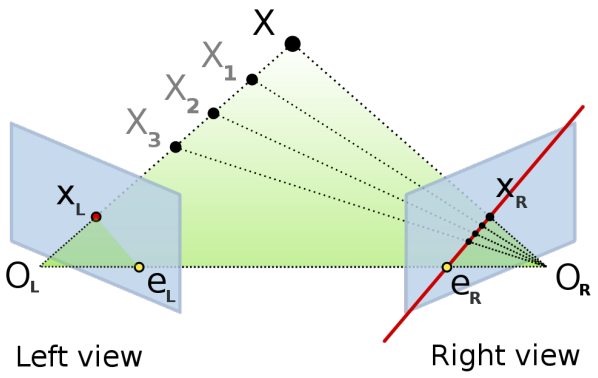


FIGURE 4 – Les plans en bleus correspondent aux plans images des deux caméras. La droite reliant les deux centres optiques est appelée *baseline*. (Image : Wikipédia)

Lorsque les deux caméras ont une orientation quelconque, les centres optiques se projettent chacun sur le plan image de l'autre caméra, en e_L et e_R respectivement. Ces deux points sont appelés *épipoles*. Dans ce cadre, pour tout point de l'espace X, le plan $(O_L\, X\, O_R)$ coupe le plan image de la caméra de gauche (resp. de droite) en une droite passant par les deux points X_L et e_L (resp. X_R et e_R) ; ces droites sont appelées *épipolaires* (car, pour tout X, elles passent par les épipoles). Une telle droite est tracée en rouge sur la figure 5.

Rectification La rectification consiste à mettre les deux plans images dans le même plan et à la même hauteur. On simule ainsi la prise de vue de la scène par deux caméras dont les deux axes optiques seraient parallèles et à la même hauteur. Lorsqu'une paire d'images est rectifiée de la sorte, on dit qu'elles sont en *géométrie épipolaire*.

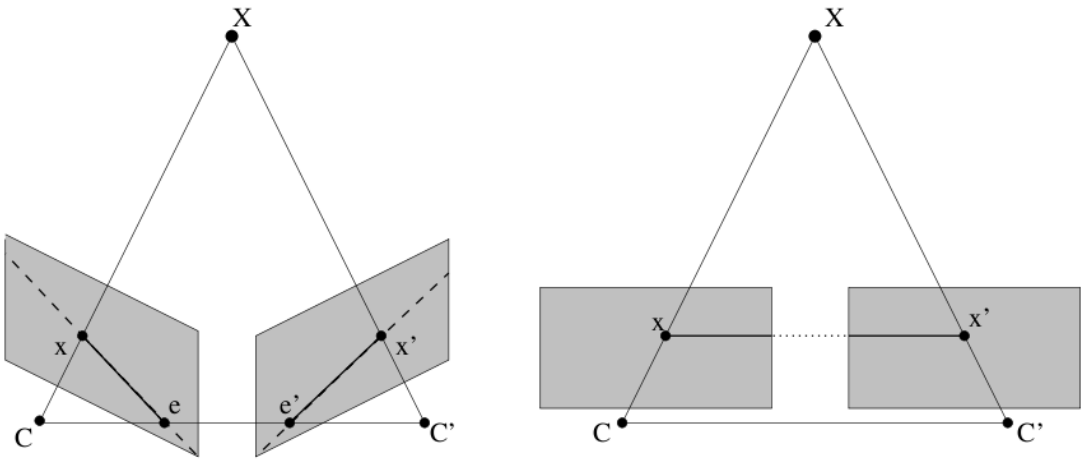


FIGURE 5 – Rectification d'une paire d'images.

Lorsque deux images sont en géométrie épipolaire, les points d'une ligne dans l'une des images ont leurs homologues dans la même ligne de l'autre image.

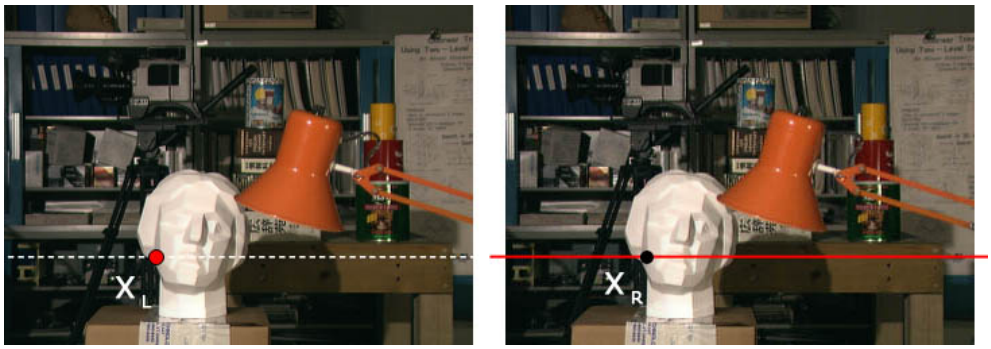


FIGURE 6 – Couple d'images Tsukuba en géométrie épipolaire : les points X_L et X_R sont situés sur la même ligne.

I.4 Phénomène d'occlusion

Une situation récurrente en stéréovision est celle de l'occlusion. Lorsqu'on regarde une scène en relief, il y a nécessairement certains points qui sont visibles sur une image mais cachés (par un obstacle) dans l'autre. Ce phénomène est appelé *occlusion*, et les pixels en question sont dits *occlus*.

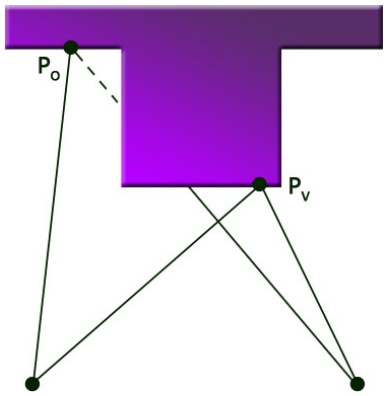


FIGURE 7 – Les caméras (ou les yeux) sont placés en O_R et O_L ; alors que le point P_V est visible par les deux caméras, seule la caméra de gauche peut "voir" le point P_O. Ce point est donc occlus.

Lorsqu'on choisit l'une des deux images comme étant l'image de référence, on distingue parfois l'occlusion (disparition d'un point dans l'autre image) de la désocclusion (apparition d'un point non visible dans l'image de référence).

I.5 Mise en correspondance

Si un point est visible à partir des deux caméras, il a un projeté sur chacun des deux plans images ; le but est, bien évidemment, de réussir à associer ces deux pixels image. L'opération qui consiste à associer à un pixel d'une image celui que l'on croit (ou espère) lui être son homologue dans l'autre image est appelée *mise en correspondance* (*matching* en anglais). Une *configuration* est alors une mise en correspondance de chaque pixel d'une image, ou, le cas échéant, la déclaration du caractère occlus de certains pixels.

Pour clarifier les idées, on peut considérer qu'une configuration f est une fonction définie sur l'ensemble des pixels de l'image de gauche (par exemple) et qui associe à un pixel un pixel de l'image de droite, ou la valeur `occlus` [2].

La mise en correspondance peut parfois être très difficile, notamment sur des surface unie (sans texture) ; dans ce cas, le choix de la bonne mise en correspondance est très délicate, car de nombreuses possibilités peuvent s'avérer crédible (notre cerveau lui-même se trompe parfois).

II Méthodes globales, méthodes locales

Les méthodes en stéréovision sont classiquement divisées en deux catégories [9, 19].

II.1 Méthodes globales

Ce type de méthode considère l'image dans son intégralité, et cherche à définir une énergie sur l'image qui quantifie à quel point une configuration est crédible (cf. section IV). Le but est alors de minimiser l'énergie en question. Cela donne alors une configuration, de laquelle on peut extraire pixel par pixel la valeur de la disparité pour chaque point.

Ces méthodes introduisent des *fonctionnelles d'énergie*, qui comportent généralement deux à quatre termes, par exemple :

$$E(f) := E_{\text{match}}(f) + E_{\text{occ}}(f) + E_{\text{reg}}(f) + E_{\text{vis}}(f)$$

où chaque terme correspond à une contrainte à laquelle une mise en correspondance doit obéir, avec f une configuration. Ces conditions concernent généralement au moins la fidélité de la mise en correspondance

2. On peut bien sûr donner d'autres définitions : f pourrait associer à un couple de pixels des deux images la valeur 1 s'ils correspondent et 0 sinon, ou encore à un pixel de l'image de référence sa disparité s'il a été mis en correpondance avec un autre pixel, et `occlus` sinon.

et la régularité (ou régularité par morceaux) de la fonction de disparité [19, 14] ; elles peuvent toutefois prendre en compte le phénomène d'occlusion [16] ou encore la *visibilité* d'un point [10].

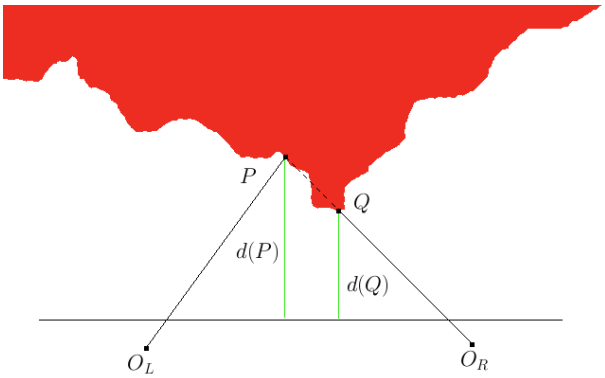


FIGURE 8 – La contrainte de visibilité est la suivante : si P est visible des deux caméras, alors il présente la même profondeur selon les deux caméras ; sinon, la profondeur du point Q vu par la seconde caméra dans la direction de P est nécessairement inférieure.

Il faut alors régler la valeur de chaque pénalité imposée ; ce genre de méthode impliquent donc un certain nombre de paramètres, ce qui les peut les rendre moins légitimes. Ces fonctionnelles d'énergie sont en général définies sous une forme qui se prête bien à la minimisation par Graph Cut [18].

L'un des plus grands désavantages de ce genre de méthodes est leur trop grande complexité de calculs.

II.2 Méthodes locales

Les méthodes locales parcourent l'image pixel par pixel afin de faire la mise en correspondance. Elles sont donc basées sur des contraintes locales. On peut citer les méthodes de *Block Matching*, qui consistent à faire la mise en correspondance bloc par bloc : pour un bloc donné dans l'image de référence, on cherche dans le voisinage de ce bloc dans l'autre image le bloc qui lui ressemble le plus (en introduisant une certaine norme).

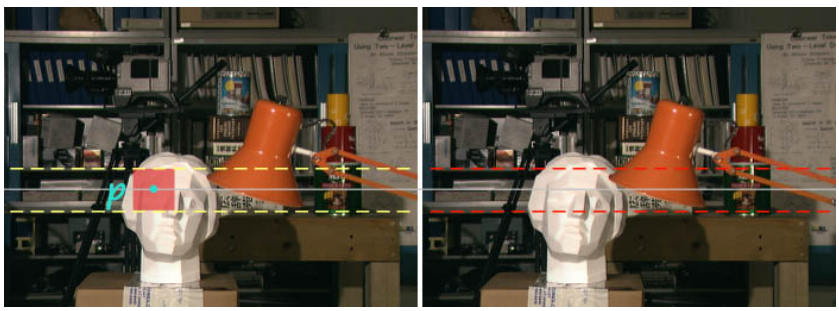


FIGURE 9 – Exemple : pour un pixel p de l'image gauche, on essaie de faire correspondre son voisinage (en rose) sur la même ligne dans l'autre image.

Il existe plusieurs méthodes pour trouver le bloc de l'image droite qui ressemble le plus ; on peut en citer deux : utilisation d'une norme (L^1 ou L^2) et NCC (*Normalized crossed correlation*).
On note I_g (resp. I_d) la fonction (ou matrice) associée à l'image gauche (resp. image droite). On note a l'entier tel que le côté du carré du bloc soit $2a+1$. Le pixel p a pour coordonnées (p_x, p_y). Enfin, on repère le bloc courant de l'image droite avec d.
Ces deux méthodes sont utilisées lorsque l'on dispose de deux signaux, dont le premier est supposé extrait du second, et que l'on cherche à retrouver le premier signal dans le second.

- La méthode de la norme consiste à minimiser la différence entre le bloc autour de p et le bloc courant : on cherche d minimisant

$$\int_{\substack{[p_x-a,p_x+a]\\ \times[p_y-a,p_y+a]}} |I_g(u,v) - I_d(u+d,v)|^2 \,\mathrm{d}u\,\mathrm{d}v$$

- La méthode NCC consiste à maximiser la quantité :

$$\int_{\substack{[p_x-a,p_x+a]\\ \times[p_y-a,p_y+a]}} I_g(u,v)\, I_d(u+d,v) \,\mathrm{d}u\,\mathrm{d}v$$

  L'idée de la NCC est la suivante : si d n'est pas la disparité recherchée, alors le produit $I_g(u,v)\, I_d(u+d,v)$ sera tantôt positif, tantôt négatif (car les deux images ne sont pas corrélées), donc la somme comporte des termes qui se compensent et n'a pas de valeur absolue élevée. En revanche, si d est la disparité recherchée, alors $I_g(u,v) = I_d(u+d,v)$ donc le produit $I_g(u,v)\, I_d(u+d,v)$ est toujours positif. La somme est donc élevée.

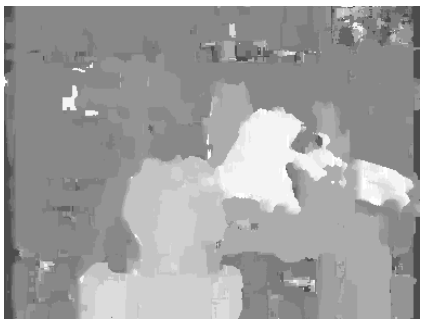


FIGURE 10 – Résultat avec une méthode NCC

Ce genre de méthode est toutefois peu fiable lorsqu'il y a peu de texture dans la scène ou en cas d'occlusion. Par ailleurs, elles détectent très mal les petites occlusions. Mais ces méthodes sont faciles à implémenter et demandent très peu de calculs.

II.3 Stratégies complémentaires

Outre ces deux grandes classes de méthodes, on peut citer quelques unes des stratégies proposées pour améliorer les résultats :

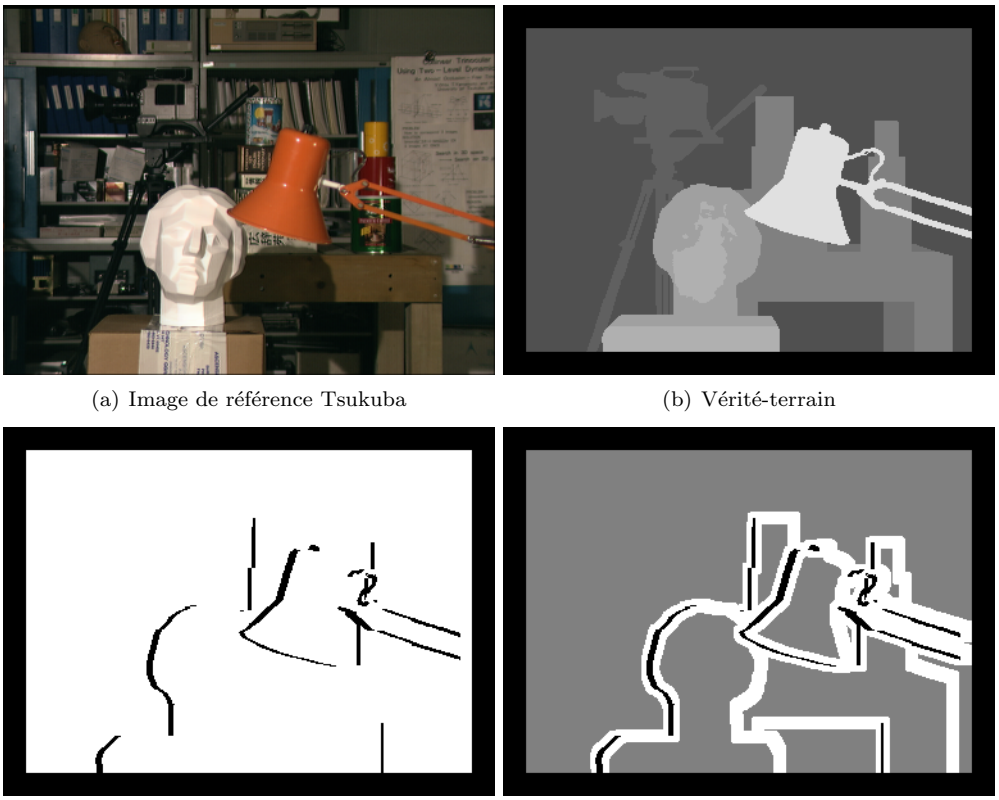
- les stratégies qui consistent à commencer par segmenter l'image en régions [22, 14] (grâce à l'algorithme du *Mean Shift* [12]), puis à mettre en correspondance des régions d'une image à l'autre ;

FIGURE 11 – Exemple de segmentation obtenue grâce à l'algorithme du *Mean Shift*. (Image : [22])

- les méthodes basées sur le *Belief Propagation*, probabilistes, qui consistent à faire propager l'information sur les pixels dont on a calculé la disparité de manière fiable (par exemple les zones texturées) vers les pixels dont on ne sait rien (ou pas grand chose) [20, 23, 14] et qui font intervenir les *champs de Markov* [7] (*Markov Random Fields* en anglais).

II.4 Le banc d'essai Middlebury

Afin de comparer les méthodes proposées par les auteurs, une équipe de chercheurs [19] a mis au point un banc d'essai [1], constitué de paires d'images stéréoscopiques dont on connaît la vérité-terrain. Les chercheurs sont alors invités à tester leur algorithme sur ces images ; les résultats sont alors classés, selon le nombre de pixels mal évalués (à un seuil près), considérés dans une certaine région de l'image, qui correspond soit aux régions non occluses, soit aux discontinuités, soit à l'image entière.



(a) Image de référence Tsukuba (b) Vérité-terrain

(c) Régions non occluses (en blanc) (d) Discontinuités de la scène (en blanc)

FIGURE 12 – Les différents masques utilisés par Middlebury pour calculer les erreurs, sur l'exemple de l'image Tsukuba.

Ce *benchmark* permet ainsi de classer les méthodes proposées par les chercheurs, qui testent leur code sur les images de la base de données et envoient leurs résultats. Cependant, on peut lui reprocher de mettre à disposition des chercheurs une banque d'images trop artificielles, qui ne reflètent donc pas la réalité ; d'exiger des algorithmes qu'ils trouvent la disparité de régions dont la profondeur est a priori non déterminable (exemple des fonds unis) ; et enfin surtout de classer les méthodes uniquement sur ces images : les meilleures méthodes selon ce classement obtenant une bien meilleure visibilité, on pourrait être tenté d'ajuster les paramètres des codes de sorte à obtenir le résultat le plus satisfaisant, puisque les vérités-terrain sont mises à disposition des chercheurs.

III Problème du flot maximal

Nous présenterons dans la suite une méthode basée sur les *Graph Cuts* (coupes dans un graphe) ; il nous faut donc commencer par exposer quelques points de théorie des graphes, notamment ceux qui touchent au problème du calcul d'un flot maximal à travers un réseau. Pour plus de précisions sur le sujet, on pourra se reporter à [11] et [5].

III.1 Graphes et réseau

Définition III.1 (Graphe) *Un* graphe $\mathcal{G}$ *est un couple* $(\mathcal{V}, \mathcal{E})$*, où* $\mathcal{V}$ *est l'ensemble des* nœuds *(ou encore appelés* sommets*) et* $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ *est l'ensemble des* arêtes *(ou* arcs*)*[3]*.*
Un graphe est dit orienté *si ses arêtes sont considérés avec leur orientation (i.e.* $a_1 := (x_1, x_2) \neq (x_2, x_1) =: a_2$ *pour* $a_1, a_2 \in \mathcal{E}$*).*

On considérera dans la suite des graphes orientés.

Définition III.2 (Réseau) *On appelle* réseau $\mathcal{N}(s,t)$ *un graphe orienté* $\mathcal{G}$ *possédant deux sommets distincts* s *et* t*, appelés respectivement* source *et* puits*.*

Définition III.3 (Capacité) *On appelle* fonction de capacité *d'un graphe orienté* $\mathcal{G}$ *une fonction* c *définie sur les arcs de* $\mathcal{G}$ *à valeurs réelles positives. Si* $a \in \mathcal{E}$*,* $c(a)$ *est la* capacité *de* a*.*

III.2 Flot et coupe dans un graphe

Soit $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, et soit $x \in \mathcal{V}$ un nœud. On note $\delta^+(x)$ (resp. $\delta^-(x)$) l'ensemble des arcs de $\mathcal{G}$ qui aboutissent en x (resp. qui partent de x) :

$$\delta^+(x) := \{a \in \mathcal{E} \mid a = (y, x)\ ,\ y \in \mathcal{V}\} \qquad \text{et} \qquad \delta^-(x) := \{a \in \mathcal{E} \mid a = (x, y)\ ,\ y \in \mathcal{V}\}$$

Si $X \subset \mathcal{V}$, alors on pose $\delta^+(X)$ l'ensemble des arcs aboutissant à un nœud de X et $\delta^-(X)$ l'ensemble des arcs partant d'un nœud de X :

$$\delta^+(X) := \bigcup_{x \in X} \delta^+(x) \qquad \text{et} \qquad \delta^-(X) := \bigcup_{x \in X} \delta^-(x)$$

Définition III.4 (Flot) *Un* flot *dans le réseau* $\mathcal{N}(s,t)$ *est une fonction réelle définie sur l'ensemble des arcs du graphe associé à* $\mathcal{N}(s,t)$ *qui vérifie la propriété suivante :*

$$\forall x \in \mathcal{V} \setminus \{s,t\}\ ,\ \sum_{a \in \delta^+(x)} f(a) = \sum_{a \in \delta^-(x)} f(a)$$

Cette propriété est une propriété de conservation *du flot, encore appelée loi de Kirchhoff.*
Un flot est dit réalisable *s'il vérifie en plus une contrainte de* capacité *:*

$$\forall a \in \mathcal{E}\ ,\ 0 \leq f(a) \leq c(a)$$

où c *est la fonction de capacité du réseau* $\mathcal{N}(s,t)$*.*

La condition de conservation du flot nous permet de montrer que

$$\sum_{x \in \delta^+(s)} f(x) - \sum_{x \in \delta^-(s)} f(x) = \sum_{x \in \delta^-(t)} f(x) - \sum_{x \in \delta^+(t)} f(x)$$

Cette quantité commune est appelée *valeur* du flot f, notée $\mathrm{val}(f)$. On définit alors un *flot maximal* comme un flot dont la valeur est maximale parmi tous les flots.

3. *Vertices* et *edges* en anglais.

Définition III.5 (Coupe) *Soit $\mathcal{N}(s,t)$ un réseau. Soit $X \subset \mathcal{V}$. Si $s \in X$ et $t \notin X$, alors $\delta^+(X) \subset \mathcal{E}$ est appelé* coupe *dans $\mathcal{N}(s,t)$.*

Si $K \subset \mathcal{E}$ est une coupe dans $\mathcal{N}(s,t)$, on définit la *capacité* de K, notée $\text{cap}(K)$, la somme des capacités des arcs dans K :

$$\text{cap}(K) := \sum_{a \in K} c(a)$$

Une coupe est alors dite *minimale* s'il n'existe pas de coupe de capacité plus petite.

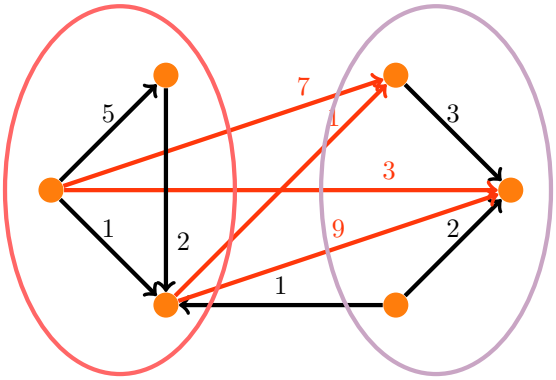


FIGURE 13 – Exemple de coupe dans un graphe (en rouge) ; la capacité de cette coupe est égale à $7+1+3+9=20$.

Lemme III.1 *Soit $\mathcal{N}(s,t)$ un réseau, et $X \subset \mathcal{V}$ tel que $s \in X$ et $t \notin X$. Alors $\delta^+(X)$ est une coupe dans $\mathcal{N}(s,t)$ et pour tout flot f dans $\mathcal{N}(s,t)$, on a :*

$$\text{val}(f) \leq \text{cap}(\delta^+(X))$$

Corollaire III.1 *Soit f un flot et K une coupe dans $\mathcal{N}(s,t)$. Si $\text{val}(f) = \text{cap}(K)$, alors f est un flot maximal et K une coupe minimale.*

Ce dernier corollaire nous montre à quel point calculer le flot maximal à travers un réseau et trouver une coupe minimale sont deux problèmes liés. Le paragraphe suivant précise ce résultat.

III.3 Théorème de Max-Flow/Min-Cut

Définition III.6 (Chaîne) *Soit $\mathcal{G}$ un graphe orienté. Une chaîne*[4] *P est une suite d'arcs consécutifs, considérés sans leur orientation. Une chaîne P est dite* issue *de $x \in \mathcal{V}$ si $P = (a_1, a_2, \ldots, a_n)$, avec $a_1 = (x,y)$; P relie x et z si $a_1 = (x,y)$ et $a_n = (w,z)$. On notera $a(P) \in \mathcal{E}$ l'ensemble des arcs de la chaîne P.*

Soit P un chemin dans le graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ et $a(P)$ l'ensemble de ses arcs. Soit $(\text{for}(P), \text{rev}(P))$ une partition de $a(P)$ définie de la manière suivante : soit $a \in a(P)$ un arc de P, $a = (x,y)$. Alors si $(x,y) \in \mathcal{E}$, $a = (x,y) \in \text{for}(P)$; sinon, $(y,x) \in \mathcal{E}$ et dans ce cas, $a = (x,y) \in \text{rev}(P)$ [5].

Soit f un flot dans le réseau $\mathcal{N}(s,t)$ de fonction de capacité c. Soit P une chaîne issue de la source s ; on lui associe le réel positif $\varepsilon(P)$ défini par :

$$\varepsilon(P) := \min\{\varepsilon(a) \mid a \in a(P)\}$$

où $\varepsilon(a)$, appelée *capacité résiduelle* de a, est donné par :

4. P pour *path* en anglais, bien que cela correspond plutôt à un chemin, c'est-à-dire une chaîne dans laquelle on prend en compte l'orientation des arcs.
5. $\text{for}(P)$ et $\text{rev}(P)$ pour *forward* et *reverse* resp.

$$\varepsilon(a) := \begin{cases} c(a) - f(a) & \text{si } a \in \text{for}(P) \\ f(a) & \text{si } a \in \text{rev}(P) \end{cases}$$

$\varepsilon(P)$ correspond au flot supplémentaire que l'on peut faire passer par la chaîne P sans violer la contrainte de capacité. Une chaîne P est alors dite *saturée* si $\varepsilon(P) = 0$, *insaturée* sinon. Une *chaîne augmentante* est une chaîne insaturée reliant la source s et le puits t.

Lemme III.2 *Soit f un flot dans un réseau $\mathcal{N}(s,t)$. S'il existe une chaîne augmentante P, alors f n'est pas un flot maximal. Plus précisément, si on définit $\tilde{f} : \mathcal{E} \longrightarrow \mathbb{R}$ par :*

$$\tilde{f}(a) := \begin{cases} f(a) + \varepsilon(P) & \textit{si } a \in \text{for}(P) \\ f(a) - \varepsilon(P) & \textit{si } a \in \text{rev}(P) \\ f(a) & \textit{sinon} \end{cases} \tag{1}$$

$\tilde{f}$ est un flot dans $\mathcal{N}(s,t)$ de valeur $\text{val}(\tilde{f}) = \text{val}(f) + \varepsilon(P)$.

Ce résultat est important pour construire un algorithme qui calculera le flot maximal, ainsi que pour le théorème qui suit :

Théorème III.1 *Soit f un flot dans un réseau $\mathcal{N}(s,t)$. On suppose qu'il n'existe pas de chaîne augmentante dans $\mathcal{N}(s,t)$. Soit $X \subset \mathcal{V}$ l'ensemble des nœuds reliés à s par une chaîne insaturée. Alors f est un flot maximal dans $\mathcal{N}(s,t)$ et $\delta^+(X)$ une coupe minimale.*

L'utilisation des chaînes augmentantes permet alors de montrer le théorème fondamental suivant :

Théorème III.2 (Max-Flow/Min-Cut) *Dans tout réseau, la valeur du flot maximal est égale à la capacité de la coupe minimale.*

III.4 Algorithme de Ford-Fulkerson (variante)

L'algorithme de Ford–Fulkerson [13] est un algorithme classique utilisé pour calculer le flot maximal dans un graphe. Le code de Kolmogorov-Zabih utilisant une variante de cet algorithme [6, 15], mieux adapté aux graphes spécifiques que l'on retrouve en vision, c'est celui-ci que nous allons présenter. Cet algorithme est essentiellement basé sur le résultat montré dans le paragraphe prédécent : un flot dans un réseau est maximal si, et seulement si, il n'existe aucune chaîne augmentante pour ce flot. Les algorithmes basés sur ce principe cherchent donc à chaque étape à trouver une chaîne augmentante, à augmenter le flot en considérant le nouveau flot donné en (1).

Soit $\mathcal{N}(s,t)$ un réseau de source s et de puits t. On va construire deux arbres sur le réseau $\mathcal{N}(s,t)$, S et T, de racines s et t respectivement, tels que S et T ne se recouvrent pas (i.e. n'ont aucun nœud en commun) et tels que toute arête reliant un parent et un enfant ne soient pas saturés. On se reportera à l'annexe pour la définition précise d'un arbre et le vocabulaire usuel associé aux arbres.

Terminologie On qualifiera de *libre* tout nœud de $\mathcal{N}(s,t)$ qui n'appartient ni à S ni à T. Un nœud est donc soit libre, soit dans S ou T ; dans le dernier cas, ils seront soit dits *actifs*, soit *inactifs*. Les nœuds actifs correspondent aux nœuds par lesquels l'arbre auxquels ils appartiennent peuvent encore croître, et les nœuds inactifs correspondent aux nœuds internes. Plus précisément, un nœud de S est inactif si tous ses enfants potentiels sont déjà des nœuds de S (*idem* pour T).

Il est important de remarquer qu'un nœud actif peut être en contact avec un nœud de l'autre arbre. Ainsi, une chaîne augmentante est obtenue dès qu'un nœud actif d'un des deux arbres détecte dans son voisinage un nœud actif de l'autre arbre (le voisinage d'un nœud x étant compris ici comme étant l'ensemble des nœuds du graphe tel qu'il existe une arête entre chacun de ces nœuds et x).

Algorithme L'algorithme considéré va itérer les phases suivantes :

1. **croissance :** les arbres S et T croissent jusqu'à ce qu'une chaîne augmentante est trouvée ;
2. **augmentation :** on augmente le flot dans cette chaîne et on retire les arêtes saturées ; on obtient alors une forêt ;

3. **adoption :** on restaure les arbres S et T

Détaillons chacune de ces phases.

Croissance Pendant la phase de croissance, les arbres S et T croissent. Leurs nœuds actifs explorent le réseau $\mathcal{N}(s,t)$ selon les arêtes insaturées et ajoutent à leurs enfants des nœuds libres. Ces nœuds deviennent alors soit actifs, soit inactifs, et dans le premier cas, ils contribuent à la croissance de l'arbre auquel ils appartiennent. Si un nœud actif est entouré de nœuds de son arbre, alors il devient inactif. Cette phase de croissance s'achève lorsqu'un nœud actif d'un arbre rencontre un nœud actif de l'autre arbre. On a donc trouvé une chaîne augmentante (elle relie s et t, et aucun des arcs n'est saturé, car les arbres S et T ne possèdent pas d'arcs saturés).

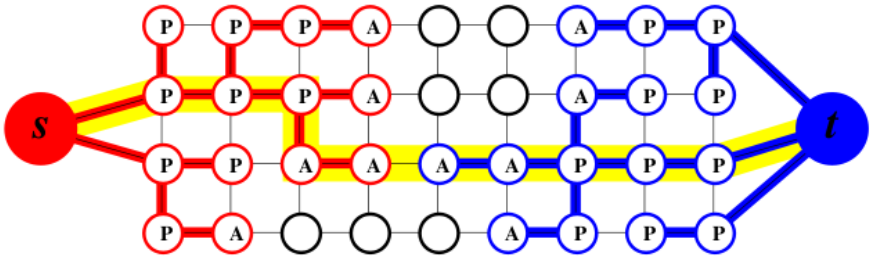


FIGURE 14 – Exemple : en rouge, l'arbre S et en bleu, l'arbre T ; les nœuds actifs et passifs sont respectivement notés A et P. (Image : [15])

Augmentation Lors de la phase d'augmentation, on commence par augmenter le flot dans la chaîne augmentante trouvée, et on sature ainsi certaines arêtes, qu'on retire. Certains nœuds deviennent alors *orphelins*, dans le sens où les arêtes les reliant à leurs parents respectifs ont été supprimés (car saturés). À l'issue de cette étape, on a donc, au lieu de deux arbres S et T, une forêt, composée de deux arbres de racines s et t et d'arbres de racines les nœuds orphelins.

Adoption Le but de la phase d'adoption est de reconstituer deux nouveaux arbres S et T, toujours de racines s et t respectivement. Il s'agit donc de trouver un parent appartenant *au même arbre* à chacun des orphelins. On explore le voisinage de l'orphelin suivant des arcs insaturés ; si l'on trouve un parent qui convient (i.e. n'importe quel nœud de l'arbre considéré dans le voisinage de l'orphelin et par une arête insaturée), alors on relie le nouveau parent et l'orphelin. Sinon, l'orphelin et ses enfants sont déclarés libres. En opérant ainsi pour chaque orphelin, on restaure la structure d'arbre pour S et T ; on notera que, puisque certains orphelins sont devenus libres, les arbres ont diminué de taille. Enfin, on rend actifs et inactifs les nœuds qu'il faut.

L'algorithme termine lorsque les arbres S et T ne peuvent plus croître (ce qui signifie qu'il n'y a plus de nœud actif) ; il n'y a donc plus de chaîne augmentante. Les arbres sont donc séparés par des arcs saturés. On a donc fait passé le flot maximal à travers ce graphe, et ainsi obtenu la coupe minimale associée, donnée par $\{S, \mathcal{V} \setminus S\}$.

Pseudo-code Avant de proposer le pseudo-code (Fig. 15) correspondant à cet algorithme, précisons qu'il est utile d'attacher à chaque nœud du graphe un drapeau contenant un certain nombre d'informations :

- $\text{tree}(x) \in \{S, T, \varnothing\}$ indique à quel arbre appartient le nœud x, ou, le cas échéant, qu'il est libre ;
- $\text{parent}(x)$ contient l'information sur le parent de x.

Par ailleurs, on définit $\text{tree\_cap}(x \to y)$ la valeur de la capacité résiduelle de l'arc (x, y) si $\text{tree}(x) = S$ et de l'arc (y, x) si $\text{tree}(x) = T$, et $\text{path}(s \to t)$ la chaîne augmentante lorsqu'elle est trouvée. Enfin, on note A l'ensemble des nœuds actifs du graphe et O l'ensemble des orphelins.

Algorithme III.1 : MAX-FLOW/MIN-CUT

initialize $S = \{s\}, T = \{t\}, A = \{s, t\}, O = \varnothing$
while true
do
- faire croître S ou T pour trouver une chaîne augmentante reliant s à t
- **if** $P = \varnothing$ **terminate**
- augmenter le flot dans P
- adopter les orphelins

procedure PHASE DE CROISSANCE
while $A \neq \varnothing$
do
- prendre un nœud $x \in A$
- **for each** voisin y de x tel que tree_cap$(x \to y) > 0$
  - **if** tree$(y) = \varnothing$
    then tree$(y) :=$ tree(x), parent$(y) := x$, $A := A \cup \{y\}$
  - **comment:** ajouter y à l'arbre en tant que nœud actif
  - **if** tree$(y) \neq \varnothing$ **and** tree$(y) \neq$ tree(x)
    then return $(P = \text{path}(s \to t))$
  - **comment:** retourner la chaîne augmentante ainsi trouvée
- retirer p de A

return $(P = \varnothing)$

procedure PHASE D'AUGMENTATION
trouver la plus petite capacité résiduelle Δ le long de la chaîne P
mettre à jour les capacités du graphe en faisant passer le flot Δ à travers P
for each arête (x, y) dans P devenue saturée
- **if** tree$(x) =$ tree$(y) = S$
  then parent$(y) := \varnothing$ **and** $O := O \cup \{y\}$
- **if** tree$(x) =$ tree$(y) = T$
  then parent$(x) := \varnothing$ **and** $O := O \cup \{x\}$

procedure PHASE D'ADOPTION
while $O \neq \varnothing$
do
- prendre un orphelin $p \in O$ et le retirer de O
- parmi ses voisins, lui chercher un parent q tel que tree$(p) =$ tree(q) **and** tree_cap > 0
- **if** p trouve un parent q
  then parent$(p) := q$
  else
  - **for all** q voisin de p tel que tree$(p) =$ tree(q)
  - **if** tree_cap$(q \to p) > 0$
    then ajouter q à l'ensemble A
    else
    - **if** parent$(q) = p$
      then ajouter q à l'ensemble des orphelins O
    - et parent$(q) := \varnothing$
  - tree$(p) := \varnothing$, $A := A \setminus \{p\}$

FIGURE 15 – Pseudo-code de l'algorithme de Max-Flow/Min-Cut [6]

IV Méthode de Kolmogorov-Zabih basée sur le Graph Cut

Nous présentons ici la méthode proposée par Kolmogorov et Zabih dans [16, 17, 15], et dont le code est accessible sur la page personnelle des auteurs. Nous avons également participé à sa mise en ligne sur le site IPOL [2].

IV.1 Notations

Soit $\mathcal{L}$ l'ensemble des pixels de l'image de gauche, $\mathcal{R}$ ceux de l'image de droite. On note $\mathcal{P} := \mathcal{L} \cup \mathcal{R}$. Les coordonnées d'un pixel p sont notées (p_x, p_y).

Assignements Soit $\mathcal{A}$ l'ensemble des paires (non ordonnés) de pixels qui peuvent potentiellement correspondre, noté $\langle p, q\rangle = \langle q, p\rangle$; dans une telle paire $\langle p, q\rangle$, on notera p le pixel issu de l'image de gauche et q celui de l'image de droite. On aura alors, en géométrie épipolaire :

$$\mathcal{A} := \{\langle p, q\rangle \mid p_y = q_y \text{ et } 0 \leq |p_x - q_x| \leq d_{\max}\}$$

où $d_{\max}$ correspond à la disparité maximale de la scène. De manière plus générale, on aura (toujours avec cette convention) :

$$\mathcal{A} := \{\langle p, q\rangle \mid y_{\min} \leq q_y - p_y \leq y_{\max} \text{ et } x_{\min} \leq q_x - p_x \leq x_{\max}\}$$

Les éléments de cet ensemble sont appelés *assignements*.
Soit f une configuration : pour tout $a \in \mathcal{A}$, $a = \langle p, q\rangle$, $f(a) = 1$ si les pixels p et q correspondent dans cette configuration (et a est alors dit *actif*), et vaut 0 sinon. On note $A(f)$ l'ensemble des assignements actifs dans f :

$$A(f) := f^{-1}(\{1\}) = \{a \in \mathcal{A} \mid f(a) = 1\}$$

On note $A_p(f)$ l'ensemble des assignements actifs dans f impliquant le pixel p, i.e.

$$A_p(f) := \{\langle p, q\rangle \in A(f)\}$$

Une configuration f est dite *unique* si pour tout $p \in \mathcal{P}$, $\text{card}(A_p(f)) \leq 1$, ce qui signifie qu'il y a au plus un assignement actif impliquant le pixel p. En d'autres termes, pour tout pixel p, il y a au plus un pixel q qui lui correspond. On notera que les pixels p tels que $\text{card}(A_p(f)) = 0$ sont exactement les pixels occlus. On introduit également les notations suivantes :

- on désigne par $N_1(f)$ le nombre d'assignements actifs dans f et $N_0(f)$ le nombre d'assignements inactifs :
$$N_1(f) = \text{card}\,(A(f)) \quad \text{et} \quad N_0(f) = \text{card}\,(\mathcal{A} \backslash A(f))$$
Alors $N_1(f) + N_0(f) = \text{card}(\mathcal{A}) = M \cdot |y_{\max} - y_{\min}|\,|x_{\max} - x_{\min}|$ est une constante, où M désigne le nombre de pixels dans l'image de référence.
- on note $n_0(f)$ le nombre de pixels occlus dans $\mathcal{P}$ et $n_1(f)$ le nombre de pixels non occlus :
$$n_0(f) = \text{card}\{p \in \mathcal{P} \mid \text{card}(A_p(f)) = 0\} \quad \text{et} \quad \text{card}\{p \in \mathcal{P} \mid \text{card}(A_p(f)) \neq 0\}$$
On a cette fois $n_0(f) + n_1(f) = 2\,M$.

On peut dès à présent expliciter la relation entre ces quatre nombres dans le cas d'une configuration f unique. Il est clair que le nombre d'assignements actifs vaut le double du nombre de pixels non occlus (donc associés) :

$$N_1(f) = 2\,n_1(f)$$

Puisque les sommes $N_1(f) + N_0(f)$ et $n_0(f) + n_1(f)$ sont constantes, on peut écrire :

$$\begin{aligned} N_0(f) &= \text{card}(\mathcal{A}) - N_1(f) \\ &= \text{card}(\mathcal{A}) - 2\,n_1(f) \\ &= \text{card}(\mathcal{A}) - 2\,(2\,M - n_0(f)) \\ N_0(f) &= (\text{card}(\mathcal{A}) - 4\,M) + 2\,n_0(f) \end{aligned}$$

Donc finalement, $N_0(f) = 2\,n_0(f) + B$ avec $B = (\text{card}(\mathcal{A}) - 4\,M)$ une constante indépendante de la configuration.

Décalage et disparité On définit pour un assignement $a = \langle p, q\rangle$ son *décalage* (ou encore disparité) :

$$d(a) := (q_x - p_x, q_y - p_y)$$

et on note $\mathcal{A}^\alpha$ l'ensemble des assignements de décalage α :

$$\mathcal{A}^\alpha := \{a \in \mathcal{A} \mid d(a) = \alpha\}$$

Ainsi, si l'assignement $a = \langle p, q\rangle$ est actif dans la configuration f (c'est-à-dire que p et q sont mis en correspondance), de décalage $d(a)$, la disparité du pixel p vaut également $d(a)$.

IV.2 Fonctionnelle d'énergie

La configuration recherchée vérifie au moins trois conditions :
- **fidélité :** si deux pixels sont mis en correspondance, ils doivent avoir une intensité similaire ;
- **occlusion :** il doit y avoir le moins de pixels occlus que possible ;
- **régularité :** la disparité doit être régulière par morceaux car la scène l'est.

On va définir ainsi pour toute configuration f une énergie $E(f)$, qui mesurera à quel point f s'éloigne de ces trois conditions.

Par convention, on associe à une configuration non-unique une énergie infinie (en ajoutant ∞ pour tout p tel que $\mathrm{card}(A_f(p)) > 1$) et si f est une configuration unique, on lui associe l'énergie de la forme :

$$E(f) := E_{\text{données}}(f) + E_{\text{régularité}}(f)$$

qu'on réécrira :

$$E(f) = E_{\text{fidélité}}(f) + E_{\text{occlusion}}(f) + E_{\text{régularité}}(f)$$

où chaque terme correspondra aux trois conditions citées plus haut.

Le terme de données On écrit le terme de données sous la forme suivante :

$$E_{\text{données}}(f) := \sum_{a\in\mathcal{A}} D_a(f(a))$$

où, si $a = \langle p, q\rangle$,

$$D_a(f(a)) = \begin{cases} D(a) & \text{si } f(a) = 1 \\ K & \text{sinon} \end{cases}$$

avec K une constante et $D(a) = (I(p) - I(q))^2$ par exemple, où $I(p)$ est l'intensité du pixel p. Réécrivons ce terme en séparant les assignements actifs et inactifs :

$$\begin{aligned} E_{\text{données}}(f) &= \sum_{a\in A(f)} D_a(f(a)) + \sum_{a\notin A(f)} D_a(f(a)) \\ &= \sum_{a\in A(f)} D(a) + \sum_{a\notin A(f)} K \\ &= \sum_{a\in A(f)} D(a) + N_0(f)\cdot K \\ &= \sum_{a\in A(f)} D(a) + (2\,n_0(f) + B)\cdot K \\ &= \sum_{a\in A(f)} D(a) + 2\,n_0(f)\cdot K + K\,B \\ E_{\text{données}}(f) &= E_{\text{fidélité}}(f) + E_{\text{occlusion}}(f) + K\,B \end{aligned}$$

en posant

$$E_{\text{fidélité}}(f) := \sum_{a\in A(f)} D(a) \quad \text{et} \quad E_{\text{occlusion}}(f) := \sum_{\substack{p\in\mathcal{P} \\ \mathrm{card}(A_p(f))=0}} 2\,K$$

Le terme $E_{\text{fidélité}}$ mesure la résultante des différences d'intensité entre deux pixels correspondants. Dans un cas idéal, il faudrait évidemment que deux pixels qui sont mis en correspondance aient exactement la même intensité ; dans ce cas, $E_{\text{fidélité}}(f)$ serait nul. Mais il existe des situations dans lesquelles ce n'est pas possible (par exemple, en cas de présence de reflets qui apparaîtraient dans une image mais pas dans l'autre, ou si la prise des deux images n'est pas simultanées, déplacement des ombres). Dans ce cas, on pénalise d'autant plus que la différence est grande (si elle est trop grande, il y a de fortes chances pour que la mise en correspondance soit fausse). On pourra se reporter à la partie IV.7 pour plus de détails sur la mesure D choisie.

Le terme d'occlusion $E_{\text{occlusion}}(f)$ compte le nombre de pixels occlus : il impose pour chaque pixel considéré comme occlu dans la configuration f la pénalité $2\,K$. Dans [16], K est fixé à $5\,\lambda$, où λ est un paramètre de la méthode.

Le terme de régularité Quand on fait l'hypothèse qu'une image est régulière par morceaux, on s'attend à ce qu'une bonne configuration soit telle que, si deux pixels sont proches, alors leur disparité est la même, tout en autorisant tout de même des discontinuités. Pour cela, il faut commencer à introduire une notion de *système de voisinage* sur une image. Un système de voisinage sera donc un ensemble :

$$\mathcal{N} \subset \{\{a_1, a_2\} \mid a_1, a_2 \in \mathcal{A}\}.$$

On le choisit de telle sorte que deux assignements a_1 et a_2 sont voisins si ils ont même disparité et s'ils impliquent des pixels voisins (dans le sens adjacent horizontalement et verticalement).

On définit alors le terme de régularité de la manière suivante :

$$E_{\text{régularité}}(f) := \sum_{\{a_1,a_2\}\in\mathcal{N}} V_{a_1,a_2} \cdot T(f(a_1) \neq f(a_2))$$

où T désigne la fonction indicatrice (qui vaut 1 si son argument est vrai, 0 sinon). Dans ce cas-là, si les pixels adjacents ont la même disparité, ce terme est nul, autrement, il est positif. En fait, lorsque deux assignements présentent une même décalage, ils contribuent à la somme uniquement si l'un est actif et l'autre pas, c'est-à-dire en particulier que s'ils sont adjacents, la configuration f ne les a pas mis en correspondance, bien qu'ils soient proches et de même disparité ; en particulier, l'un des deux assignements s'est vu attribuer une autre disparité ou implique un pixel occlus.

Dans [16], pour deux assignements $a_1 = \langle p, q\rangle$, $a_2 = \langle r, s\rangle$, V_{a_1,a_2} est choisi de la manière suivante :

$$V_{a_1,a_2} = \begin{cases} 3\,\lambda & \text{si } \max(|I(p) - I(r)|, |I(q) - I(s)|) < 8 \\ \lambda & \text{sinon} \end{cases}$$

Cette pénalité est définie de telle sorte que l'on pénalise davantage si la différence de couleur entre les pixels des deux assignements est petite ; plus précisément, si p et q sont voisins, mais de couleur différente (suffisamment pour que l'œil soit capable de les distinguer, c'est-à-dire de différence d'intensité supérieure à 8), on pénalise légèrement moins, car il est possible qu'il s'agisse d'une discontinuité physique dans la scène.

Il faut remarquer que, généralement, le terme de régularité est plutôt écrit en terme de gradient, ce qui permet de pénaliser d'autant plus que la différence de disparité est grande ; en particulier, cela permet de peu pénaliser lorsqu'il y a tellement de niveaux de profondeur possibles que deux pixels qui *presqu*'au même niveau présentent toutefois une disparité différente (situation qui risque de se produire si l'on zoome par exemple). Mais ce choix est conditionné par le fait que, écrit de cette manière, cette fonctionnelle peut être minimisée par Graph Cut (voir partie suivante).

Finalement, la fonction que l'on considère est de la forme :

$$E(f) = E_{\text{fidélité}}(f) + E_{\text{occlusion}}(f) + E_{\text{régularité}}(f) + K\,B$$

où $K\,B$ est une constante. Comme on cherche à minimiser cette fonctionnelle, on peut ignorer ce terme constant. Une fois choisie la fonction D, cette fonctionnelle ne dépend *a priori* plus que des paramètres K et λ, ou encore λ si les deux paramètres sont corrélés, comme c'est le cas dans le code [16] (voir section V pour plus de précisions concernant cette corrélation).

IV.3 Énergie et graphes

On rappelle que l'on considère une fonctionnelle d'énergie de la forme

$$E(f) = \sum_{a\in\mathcal{A}} D_a(f(a)) + \sum_{\{a_1,a_2\}\in\mathcal{N}} V_{a_1,a_2}\cdot T(f(a_1)\neq f(a_2)) + \sum_{\{\langle p,q\rangle,\langle p,r\rangle\}} \infty\cdot T(f(\langle p,q\rangle)\cdot f(\langle p,r\rangle)=1)$$

où $\mathcal{N}$ est un système de voisinage de $\mathcal{A}$, donc un sous-ensemble (fixé) de $\mathcal{A}$, et, si $a=\langle p,q\rangle$,

$$D_a(f(a)) = \left\{ \begin{array}{cc} D(a) & \text{si } f(a)=1 \\ K & \text{sinon} \end{array} \right.$$

On peut encore l'écrire de la manière suivante :

$$E(f) = \sum_{a\in\mathcal{A}} D_a(f(a)) + \sum_{\{a_1,a_2\}\in\mathcal{A}\times\mathcal{A}} V^{\mathcal{N}}_{a_1,a_2}\cdot T(f(a_1)\neq f(a_2)) + \sum_{\{\langle p,q\rangle,\langle p,r\rangle\}} \infty\cdot T(f(\langle p,q\rangle)\cdot f(\langle p,r\rangle)=1)$$

où $V^{\mathcal{N}}_{a_1,a_2} = \left\{ \begin{array}{cc} V_{a_1,a_2} & \text{si } \{a_1,a_2\}\in\mathcal{N} \\ 0 & \text{sinon} \end{array} \right.$.
Si on ordonne les éléments de $\mathcal{A} = \{a_{1\,1},\ldots,a_{d\,1},\cdots,a_{1\,p},\ldots,a_{d\,p}\}$ (ordre lexicographique pour une certaine relation d'ordre sur les pixels), cette fonctionnelle est de la forme

$$\begin{aligned} E(f) = \sum_{i=1}^{d}\sum_{j=1}^{p} D_{a_{i\,j}}(f(a_{i\,j})) + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{p}\sum_{k=1}^{d}\sum_{l=1}^{p} V^{\mathcal{N}}_{a_{i\,j},a_{k\,l}}\cdot T(f(a_{i\,j})\neq f(a_{k\,l})) \\ + \sum_{i=1}^{d}\sum_{j=1}^{p}\sum_{l=1}^{p}\infty\cdot T(f(a_{i\,j})\cdot f(a_{i\,l})=1) \end{aligned}$$

Définition IV.1 (Classe $\mathcal{F}^2$) *On définit les fonctions de la classe $\mathcal{F}^2$ comme étant les fonctions E pouvant être écrites comme sommes de fonctions d'une variable binaire et de deux variables binaires :*

$$E(x_1,\ldots,x_n) = \sum_i E^i(x_i) + \sum_{i<j} E^{i,j}(x_i,x_j)$$

La fonctionnelle d'énergie que l'on considère fait partie de cette classe de fonctions.

Nous allons définir la représentation d'une fonction de $\mathcal{F}^2$ par un graphe. Soit $\mathcal{G}=(\mathcal{V},\mathcal{E})$ un graphe orienté, de nœuds $\mathcal{V}=\{v_1,\ldots,v_n,s,t\}$, avec s et t deux nœuds distincts. Toute coupe $(\mathcal{V}^s,\mathcal{V}^t)$ dans ce graphe peut être décrit par n variables binaires $x_1,\ldots,x_n$ correspondant aux nœuds de $\mathcal{V}\setminus\{s,t\}$, de sorte que $x_i=0$ si $v_i\in\mathcal{V}^s$, $x_i=1$ sinon. Le poids d'une telle coupe est alors une fonction de n variables binaires $C(x_1,\ldots,x_n)$.

On peut généraliser cette notion en ne considérant qu'un sous-ensemble

$$\mathcal{V}_0 = \{v_1,\ldots,v_k\}\subset\mathcal{V}\setminus\{s,t\}$$

et n'introduire que des variables binaires associées aux nœuds de cet ensemble. Il n'y a alors pas unicité de la coupe correspondant à une configuration donnée $(x_1,\ldots,x_k)$.

Définition IV.2 (Représentation par un graphe) *Une fonction E de n variables binaires est dite* représentable par un graphe *s'il existe un graphe $\mathcal{G}=(\mathcal{V},\mathcal{E})$ possédant deux sommets distincts s et t et un sous-ensemble de sommets $\mathcal{V}_0=\{v_1,\ldots,v_n\}\subset\mathcal{V}\setminus\{s,t\}$ tels que, pour toute configuration $x_1,\ldots,x_n$, la valeur $E(x_1,\ldots,x_n)$ soit égale à la plus petite capacité des coupes $(\mathcal{V}^s,\mathcal{V}^t)$, définie par : $v_i\in\mathcal{V}^s$ si $x_i=0$ et $v_i\in\mathcal{V}^t$ si $x_i=1$ pour $i\in[\![1,n]\!]$, plus une constante.*
On dit que E est exactement représentée *par $\mathcal{G}$ et $\mathcal{V}_0$ si cette constante est nulle.*

Le lemme suivant est alors une conséquence immédiate de cette définition :

Lemme IV.1 *Si E est représentable par le graphe $\mathcal{G}$ et un sous-ensemble $\mathcal{V}_0$, alors on peut minimiser E en calculant la coupe minimale sur $\mathcal{G}$.*

On cherche à caractériser les fonctions de la classe $\mathcal{F}^2$ qui sont représentables par un graphe.

Théorème IV.1 (Théorème $\mathcal{F}^2$) *Soit E une fonctions de n variables binaires de la classe $\mathcal{F}^2$:*

$$E(x_1, \ldots, x_n) = \sum_i E^i(x_i) + \sum_{i<j} E^{i,j}(x_i, x_j)$$

Alors E est représentable par un graphe si, et seulement si, tous les termes $E^{i,j}(x_i, x_j)$ satisfont la condition suivante :

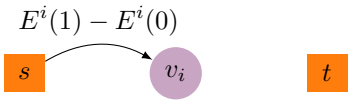
$$E^{i,j}(0,0) + E^{i,j}(1,1) \leq E^{i,j}(0,1) + E^{i,j}(1,0)$$

Une telle fonction est dite régulière.

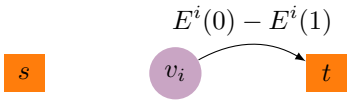
Il faut remarquer que l'écriture d'une fonction de $\mathcal{F}^2$ en somme de fonctions d'une et de deux variables binaires n'est pas unique. Mais on peut montrer que le caractère régulier d'une fonction de $\mathcal{F}^2$ ne dépend pas de sa représentation.

Détaillons la construction d'un tel graphe pour une fonction régulière. Il s'agit de représenter chacun des termes de la fonction par un graphe. Il n'y a pas unicité de la représentation, nous présenterons ici celles qui comportent le moins d'arcs.
On commence par les termes E^i à une seule variable binaire. Si $E^i(0) < E^i(1)$, alors on construit l'arc (s, v_i) de poids $E^i(1) - E^i(0)$, sinon, on construit l'arc (v_i, t) de poids $E^i(0) - E^i(1)$ (les fonctions d'une seule variable sont toujours représentables par des graphes). On vérifie immédiatement qu'ainsi construit, le graphe représente bien la fonction ; le premier cas, la constante vaut $-E^i(0)$ et dans le second, $-E^i(1)$. Le graphe représentant la somme $\sum_i E^i$ est alors le graphe de nœuds $\{v_1, \ldots, v_n, s, t\}$ avec les arcs construits comme précédemment. La constante vaut alors

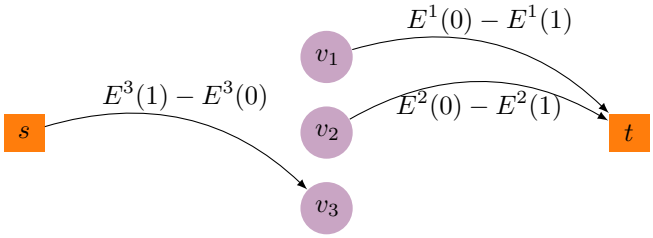
$$-\sum_i [E^i(0) \cdot T(E^i(0) < E^i(1)) + E^i(1) \cdot T(E^i(0) > E^i(1))]$$



(a) Cas où $E^i(0) < E^i(1)$



(b) Cas où $E^i(0) > E^i(1)$



(c) Somme de fonctions d'une seule variable binaire

FIGURE 16 – (a) et (b) : Représentation par un graphe d'une fonction d'une seule variable binaire. (c) : Exemple de représentation d'une somme de telles fonctions, de constante $C = -[E^1(1) + E^2(1) + E^3(0)]$. Si on considère la coupe $(\{s, v_1\}, \{t, v_2, v_3\})$, la coupe est de poids $E^1(0) + E^2(1) + E^3(1) + C$.

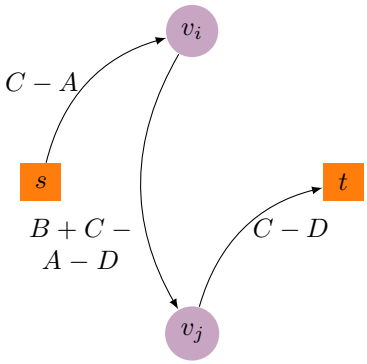
On considère maintenant le terme $E^{i,j}$ fonction de deux variables binaires. En représentant ce terme dans un tableau comme suit :

$$E^{i,j} = \begin{array}{|c|c|} \hline E^{i,j}(0,0) & E^{i,j}(0,1) \\ \hline E^{i,j}(1,0) & E^{i,j}(1,1) \\ \hline \end{array}$$

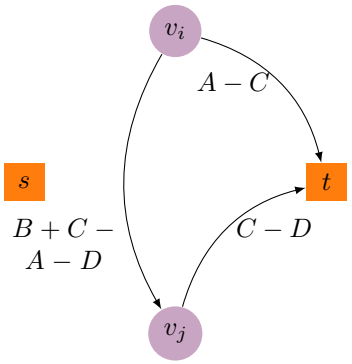
on peut décomposer $E^{i,j}$ de la manière suivante, en posant $A = E^{i,j}(0,0)$, $B = E^{i,j}(0,1)$, $C = E^{i,j}(1,0)$ et $D = E^{i,j}(1,1)$:

$$E^{i,j} = A + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline C-A & C-A \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & D-C \\ \hline 0 & D-C \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & B+C-A-D \\ \hline 0 & 0 \\ \hline \end{array}$$

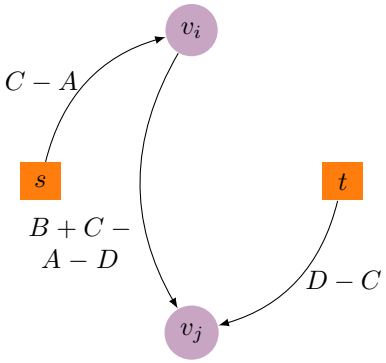
Le premier terme est la fonction constante égale à $E^{i,j}(0,0)$, donc, comme la représentation par graphe se fait à un terme additif près, on peut l'ignorer. Le second terme correspond à une fonction à deux variables binaires qui vaut 0 pour $(x_i, x_j) \in \{(0,0),(0,1)\}$ et $E^{i,j}(1,0) - E^{i,j}(0,0)$ pour $(x_i, x_j) \in \{(1,0),(1,1)\}$, elle ne dépend donc que de x_i : on reprend la construction introduite pour les fonctions à une seule variable binaire. On construit de la même manière les arcs pour le troisième terme, qui ne dépend que x_j. Enfin, le dernier terme est une fonction qui vaut $E^{i,j}(0,1) + E^{i,j}(1,0) - E^{i,j}(0,0) - E^{i,j}(1,1)$ si $(x_i, x_j) = (0,1)$, 0 sinon, donc on construit l'arc (v_i, v_j), de poids $B + C - A - D$.



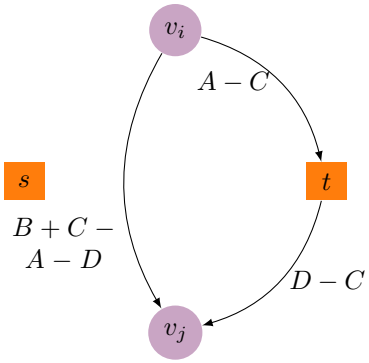
(a) Cas où $C - A > 0$ et $D - C < 0$



(b) Cas où $C - A < 0$ et $D - C < 0$



(c) Cas où $C - A > 0$ et $D - C > 0$



(d) Cas où $C - A < 0$ et $D - C > 0$

FIGURE 17 – Les différents cas de figure possibles, suivant les signes de $A - C$ et $C - D$.

On pourra consulter [15, 18] pour plus de précisions sur les fonctions de classe $\mathcal{F}^2$ (et même $\mathcal{F}^3$, qui ajoutent les fonctions de trois variables binaires).

On revient à la fonctionnelle d'énergie de la partie IV.2. Pour tout assignement a_1, a_2, elle vérifie :

$$V^{\mathcal{N}}_{a_1,a_2}(1,1) = V^{\mathcal{N}}_{a_1,a_2}(0,0) = 0$$

et $V^{\mathcal{N}}_{a_1,a_2}(0,1)$ et $V^{\mathcal{N}}_{a_1,a_2}(1,0)$ sont positifs. Donc cette fonctionnelle satisfait bien la condition de régularité du théorème $\mathcal{F}^2$: elle est représentable par un graphe.

IV.4 *α-expansion move*

Une des notions-clés de la méthode de Kolmogorov–Zabih est celle des *expansion moves*. Elle constitue le cœur de la minimisation de la fonctionnelle d'énergie.

Soit p un pixel de l'image de gauche $\mathcal{L}$; pour une configuration unique f donnée, s'il existe $q \in \mathcal{R}$ tel que $\langle p, q\rangle \in A(f)$, on définit $f_p := d(\langle p, q\rangle)$.

Définition IV.3 (*α-expansion move*) *Soit α une disparité donnée. f' est un* α-expansion move *de f si pour tout $p \in \mathcal{L}$ tel que* $\mathrm{card}(A_p(f')) = 1$, $f'_p = f_p$ *ou* $f'_p = \alpha$.

Un *α-expansion move* de f est donc une nouvelle configuration, où certains pixels de l'image de gauche se sont soit vus attribuer la disparité α, soit retirer leur affectation [8]. L'ensemble des pixels de disparité augmente *potentiellement* ; l'ensemble des pixels de disparité différente de α diminue quant à lui de manière certaine.

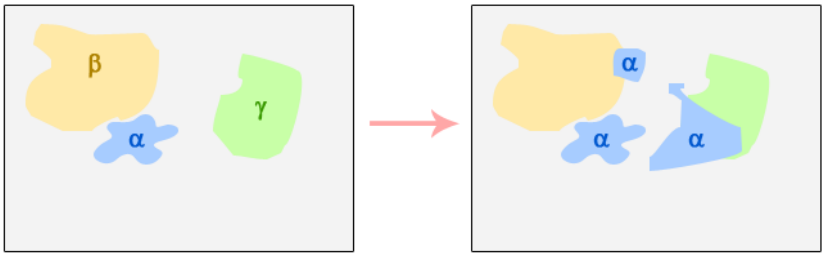


Figure 18 – On part d'une carte de disparité donnée, et on étend l'ensemble de pixels de disparité α ou on rend occlus certains pixels.

En vue d'une représentation de la fonctionnelle d'énergie par un graphe, on peut encoder un *α-expansion move* par le vecteur binaire $y = \{y_a \mid a \in \mathcal{A}^\circ \cup \mathcal{A}^\alpha\}$, tel que, pour la configuration f correspondante,

$$\begin{aligned} \forall\, a \in \mathcal{A}^\circ \quad f(a) &= 1 - y_i \\ \forall\, a \in \mathcal{A}^\alpha \quad f(a) &= y_a \\ \forall\, a \notin \mathcal{A}^\circ \cup \mathcal{A}^\alpha \quad f(a) &= 0 \end{aligned}$$

Le principe est le suivant : étant donnée une configuration f, on cherche, pour un α donné, à trouver le *α-expansion move* de f d'énergie minimale. Les α sont pris dans l'ensemble des disparités acceptables $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$.

Algorithme IV.1 : Minimisation d'une énergie par α-expansion move

initialize Soit f° une configuration
for all $\alpha \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$
do
- réaliser un *α-expansion move*
- calculer l'énergie de la configuration f obtenue
- **if** $E(f) < E(f^\circ)$
  - **then** $f^\circ := f$

return (f°)

Un tel cycle est appelé *itération*. Le code permet de choisir le nombre d'itérations voulues, sachant que [15] estime qu'au bout de cinq itérations, les résultats convergent (i.e. des itérations supplémentaires ne modifient plus le résultat), mais qu'en pratique trois itérations suffisent.

Il s'agit donc de minimiser l'énergie suivant une classe de perturbations précises. Cependant, il est impensable de trouver le *α-expansion move* optimal pour un α donné en les réalisant un par un, car si l'image est de taille M, il y en aurait 2^M. La construction d'un graphe va nous permettre de contourner cette difficulté, car l'énergie que l'on considère se prête bien à une minimisation par Graph Cuts [18].

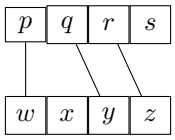
IV.5 Construction du graphe

Soit f° une configuration. On rappelle que $\mathcal{A}(f^\circ)$ est l'ensemble des assignements actifs de f°, c'est-à-dire l'ensemble des couples de pixels qui correspondent selon la configuration f°. Parmi ceux-là, $\mathcal{A}^\alpha \cap \mathcal{A}(f^\circ)$ est l'ensemble des assignements de disparité α. On note $\mathcal{A}^\circ = \mathcal{A}(f^\circ) \setminus (\mathcal{A}^\alpha \cap \mathcal{A}(f^\circ))$. Dans ce cadre, un *$\alpha$-expansion move* consiste à modifier l'ensemble $\mathcal{A}^\alpha$ en rendant inactif certains assignements dans $\mathcal{A}^\circ$ et en affectant à l'un des deux pixels en question la disparité α. On notera que, ce faisant, on peut obtenir une configuration non unique (dans le sens donné dans le début de cette section).

On a montré dans la section IV.3 que l'énergie fonctionnelle E que l'on considère était représentable par un graphe. Dans ce cadre, si $y = \{y_a \mid a \in \mathcal{A}^\circ \cup \mathcal{A}^\alpha\}$ encode un α-expansion move f de f°, et $x = \{f(a) \mid a \in \mathcal{A}^\circ \cup \mathcal{A}^\alpha\}$ le vecteur binaire encodant la configuration f, son énergie est donnée par $E(x)$. Trouver le *α-expansion move* qui décroît le plus l'énergie, c'est donc trouver la coupe minimale dans le graphe représentant E construit sur les assignements $\mathcal{A}^\circ \cup \mathcal{A}^\alpha$.

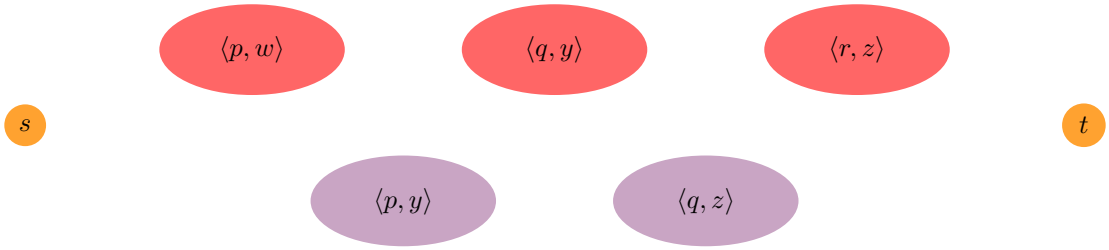
Le graphe représentant la fonctionnelle n'étant pas unique, on propose ici une construction différente de celle de la partie IV.3, proposée dans [16].

Note : Afin de faciliter la compréhension, on illustrera chaque étape de la construction par un exemple, basé sur les images 4×1 suivantes :



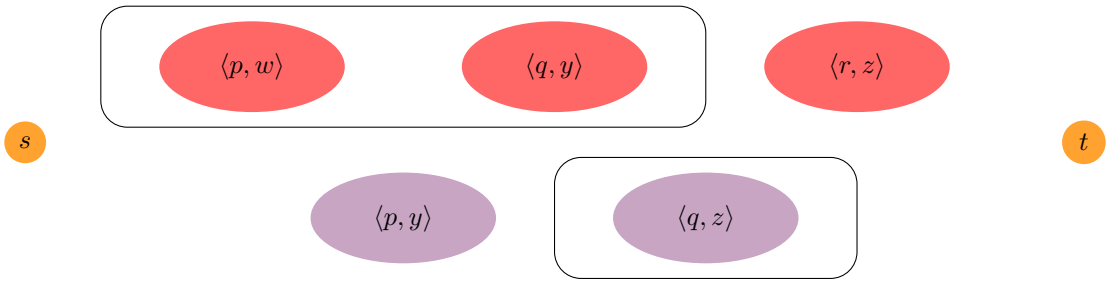
Les traits pleins représentent une mise en correspondance. La configuration f° associe donc p à w (disparité : 0), q à y et r à z (disparité : 1) ; les pixels s et x sont occlus. On choisira $\alpha = 2$.

On construit un réseau $\mathcal{N}(s,t)$ tel que s et t soient deux nœuds distincts, et tel que l'ensemble des autres nœuds (appelés *nœuds intermédiaires*) soit $\tilde{A} = \mathcal{A}^\circ \cup \mathcal{A}^\alpha$ (union disjointe par définition de $\mathcal{A}^\circ$), pour une configuration initiale f° donnée. Les nœuds intermédiaires sont donc d'une part les assignements actifs de f° (en d'autres termes, les couples de pixels mis en correspondance) de disparité différente de α, et d'autre part les assignements actifs de disparité α et les assignements inactifs de décalage α.



En rouge, les assignements actifs de disparité différentes de $\alpha = 2$ ($\mathcal{A}^\circ$), en bleu, les assignements (actifs et inactifs) de disparité α ($\mathcal{A}^\alpha$).

Rendre actif l'un de ces derniers revient à augmenter le nombre d'assignements actifs de disparité α. On notera que, pour un pixel p donné, il ne peut apparaître au plus qu'une fois dans $\mathcal{A}^\circ$ (par unicité de f°) et au plus une fois dans $\mathcal{A}^\alpha$ (par définition de $\mathcal{A}^\alpha$).



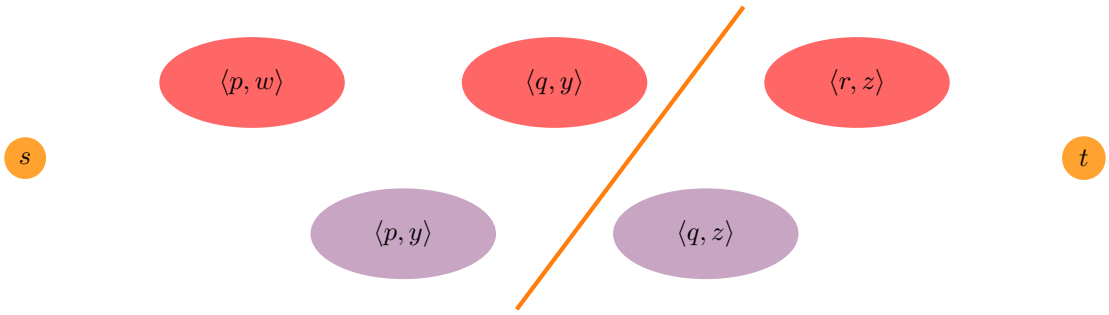
On réalise un *α-expansion move* : on conserve les assignements actifs du rectangle supérieur et on prend les assignements de disparité α du rectangle inférieur.

Dans ce cadre, étant donnée une partition des nœuds du graphe $(\mathcal{V}^s, \mathcal{V}^t)$, avec $s \in \mathcal{V}^s$ et $t \in \mathcal{V}^t$, et $\mathcal{C}$ la coupe correspondante, on définit le *α-expansion move* correspondant $f^{\mathcal{C}}$ de la manière suivante : les assignements actifs qui sont dans la même classe que s restent actifs dans la configuration $f^{\mathcal{C}}$, tandis que les assignements de décalage α qui sont dans la classe $\mathcal{V}^t$ deviennent actifs

$$\forall a \in \mathcal{A}^\circ \quad f_a^{\mathcal{C}} = \begin{cases} 1 & \text{si } a \in \mathcal{V}^s \\ 0 & \text{si } a \in \mathcal{V}^t \end{cases}$$

$$\forall a \in \mathcal{A}^\alpha \quad f_a^{\mathcal{C}} = \begin{cases} 0 & \text{si } a \in \mathcal{V}^s \\ 1 & \text{si } a \in \mathcal{V}^t \end{cases}$$

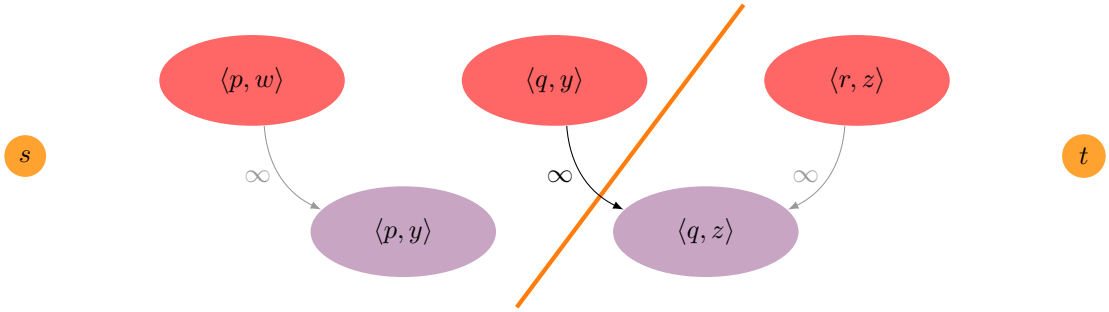
$$\forall a \notin \tilde{A}, a \in \mathcal{A}, \ f_a^{\mathcal{C}} = 0.$$



On peut donc définir la coupe correspondante ; telle qu'elle est définie, une coupe (symbolisée en orange) donne sans ambiguité l'*α-expansion move* associée.

Il faut maintenant construire les arêtes de ce graphe. On souhaite faire en sorte que le poids d'une coupe soit, à une constante près, la valeur de l'énergie de la configuration correspondante.

Soit $(\mathcal{V}^s, \mathcal{V}^t)$ une partition des nœuds du graphe, et $\mathcal{C} = \delta^+(\mathcal{V}^s)$ la coupe correspondante. On souhaite avoir une configuration unique, donc si le pixel p apparaît plusieurs fois dans le graphe, il apparaît deux fois, une fois dans un assignement actif, une fois dans un assignement inactif. Il faut donc interdire l'apparition de p dans deux nœuds actifs a et b, le premier de disparité α et le second de disparité différente de α ; si c'était le cas, et si l'arc (b, a) existait, alors il serait dans la coupe. On construit donc l'arc (b, a) et on lui attribue une capacité infinie ; ainsi, si $f^{\mathcal{C}}$ n'était pas unique, alors la coupe correspondante serait de capacité infinie et l'énergie correspondante également, ce qui était la convention adoptée.

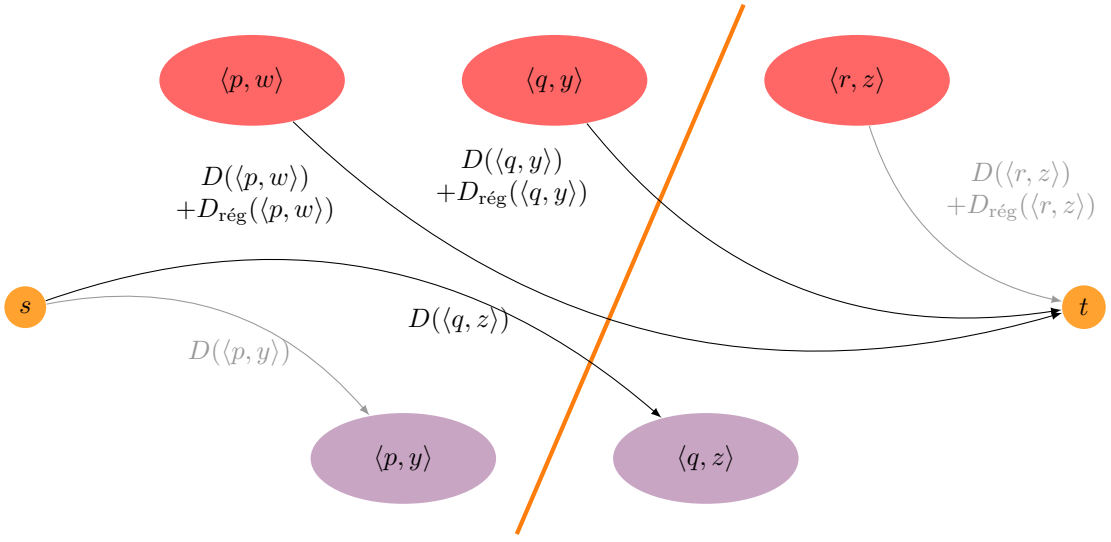


Condition d'unicité : les arcs qui contribuent au poids de la coupe sont ceux qui en font partie (ici, ce sont ceux qui coupent la droite orange). Dans notre exemple, seul l'arc reliant $\langle q, y\rangle$ et $\langle q, z\rangle$ contribuera : on pénalise d'un coût infini le fait que q soit associé à deux pixels différents dans deux assignements actifs (la configuration n'est donc pas unique).
Remarque importante : si un pixel p apparaissait deux fois dans deux assignements inactifs, l'arc reliant celui dans $\mathcal{A}^\circ$ (rouge) à celui dans $\mathcal{A}^\alpha$ (bleu) ne contribuerait pas au poids de la coupe (car par définition d'une coupe, on ne garde que les arcs allant de $\mathcal{V}^s$ vers $\mathcal{V}^t$. En revanche, il contribuerait au terme d'occlusion.

Si a est un assignement actif, alors il contribue au terme de fidélité (avec un coût de $D(a)$) ainsi qu'au terme de régularité : si b est inactif de décalage $d(a)$ et si a et b ont un pixel voisin, alors on paie $V_{a,b}$. Pour de tels b n'appartenant pas à $\tilde{A}$, la pénalité totale payée ne dépend pas de la coupe. On définit alors le *coût de régularité* :

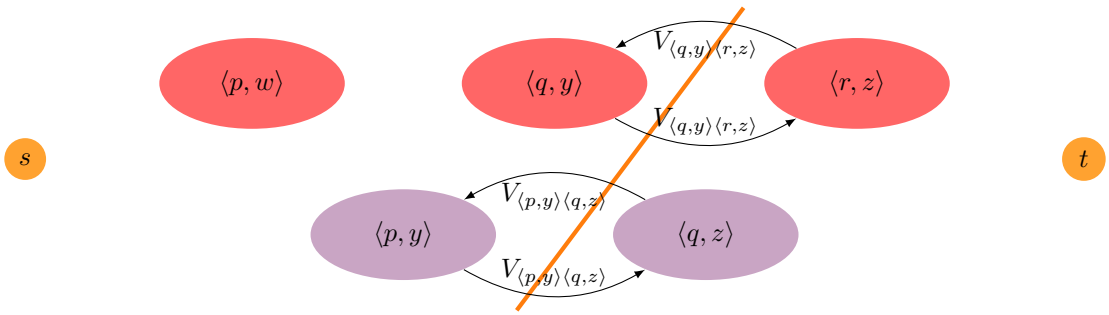
$$D_{\text{régularité}}(a) = \sum_{\substack{\{a,b\}\in\mathcal{N}\\ b\notin\tilde{A}}} V_{a,b}$$

Dans ce cas, puisque a est soit dans $\mathcal{V}^s$, soit dans $\mathcal{V}^t$, on construit les deux arcs (s, a) si $a \in \mathcal{A}^\alpha$ et (a, t) sinon, et ces deux arcs sont nécessairement dans la coupe ; on leur attribue donc un poids de $D(a) + D_{\text{régularité}}(a)$. On pourra noter que, si $a \in \mathcal{A}^\circ$, l'ensemble $\{\{a, b\} \in \mathcal{N} \mid b \notin \tilde{A}\}$ est vide et le coût de régularité est alors nul.



Terme de fidélité et première partie du terme de régularité : les trois nœuds actifs contribuent chacun au terme de fidélité (terme en D) ; par ailleurs, on peut d'ores et déjà faire apparaître leur contribution au terme de régularité avec les assignements inactifs qui n'apparaissent pas sur le graphe (cela ne concerne donc que les assignements de $\mathcal{A}^\circ$).

Si b est appartient à $\tilde{A}$, alors, comme il est de même décalage que a, il n'appartient pas à la même classe que a ; si on construit les arcs (a, b) et (b, a), l'un des deux est dans la coupe. On leur attribue donc le poids $V_{a,b}$.

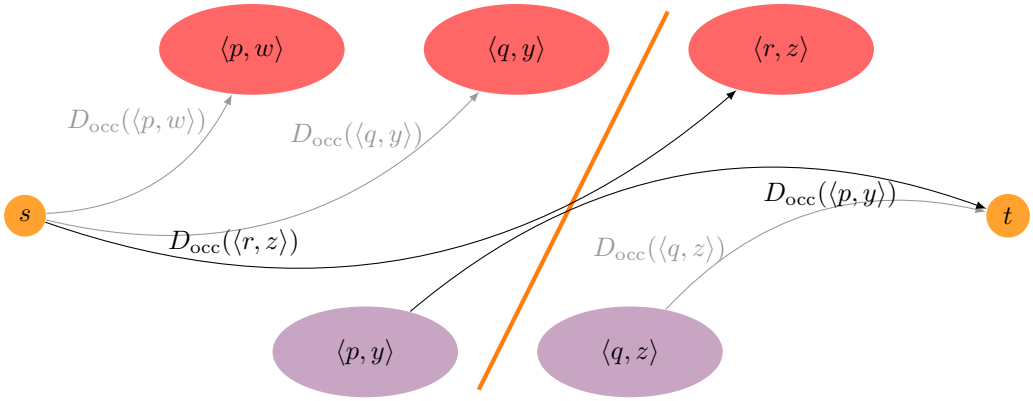


Terme de régularité (seconde partie) : $\langle q, y\rangle$ et $\langle r, z\rangle$ sont voisins, ainsi que $\langle p, y\rangle$ et $\langle q, z\rangle$. Les deux couples contribuent au terme de régularité, car dans chque cas, l'un des deux assignements est inactif. En revanche, seul un des arcs est pris en compte.

Si $a = \langle p, q\rangle$ est inactif, alors il ne contribue au terme d'occlusion que s'il n'existe aucun assignement (actif) impliquant p ou q (les pixels sont alors occlus). La pénalité correspondante vaut alors $2\,K$ si p n'apparait qu'une seule fois et q deux fois, $2\,K$ dans le cas symétrique, $4\,K$ si les deux pixels n'apparaissent qu'une seule fois chacun, et 0 sinon. On définit donc le *coût d'occlusion* :

$$D_{\text{occlusion}}(\langle p, q\rangle) = D_{\text{occlusion}}(p) + D_{\text{occlusion}}(q)$$

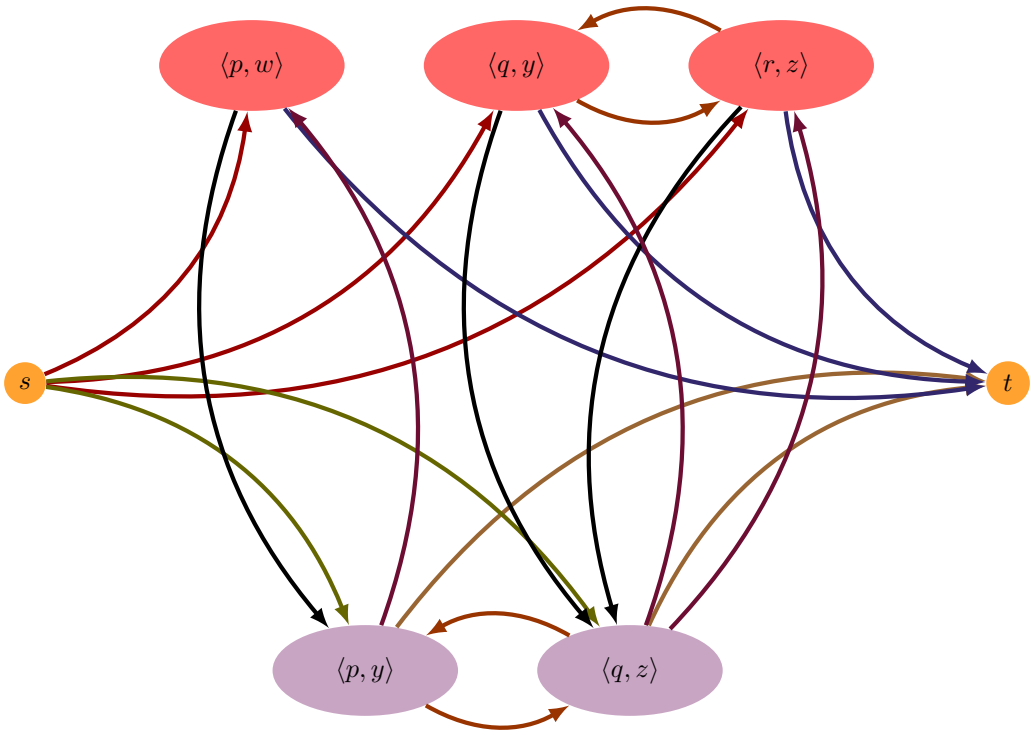
où $D_{\text{occlusion}}(p) = 2\,K$ si un pixel p n'apparaît qu'une seule fois dans le graphe, et 0 sinon. On construit alors les arcs (s, a) si $a \in \mathcal{A}^\circ$ et (a, t) sinon, auquel on attibue un poids $D_{\text{occlusion}}(a)$; par ailleurs, il faut considérer les pixels p occlus qui apparaissent dans deux assignements inactifs $a_1 \in \mathcal{A}^\circ$ et $a_2 \in \mathcal{A}^\alpha$: on construit pour cela l'arc (a_2, a_1) qu'on pondère avec $2\,K$. Il est à noter cependant que ce cas n'est jamais admis, par définition de l'*expansion move* (cela impliquerait $f_p^{\mathcal{C}} \notin \{f_p^\circ, \alpha\}$).



Terme d'occlusion : on n'a pas représenté ici les arcs de poids $2\,K$ (qui font de pair avec ceux de poids infini, cf commentaire de la figure correspondante).

On peut récapituler les arêtes et leur poids dans le tableau suivant :

arc	poids	pour
(s, a)	$D_{\text{occlusion}}(a)$	$a \in \mathcal{A}^\circ$
(a, t)	$D_{\text{occlusion}}(a)$	$a \in \mathcal{A}^\alpha$
(a, t)	$D(a)+D_{\text{régularité}}(a)$	$a \in \mathcal{A}^\circ$
(s, a)	$D(a)$	$a \in \mathcal{A}^\alpha$
(a_1, a_2) (a_2, a_1)	V_{a_1,a_2}	$\{a_1, a_2\} \in \mathcal{N}$, $a_1, a_2 \in \tilde{A}$
(a_1, a_2)	∞	$p \in \mathcal{P}$, $a_1 \in A^\circ$ $a_2 \in \mathcal{A}^\alpha$, $a_1, a_2 \in A_p(f)$
(a_2, a_1)	$2\,K$	$p \in \mathcal{P}$, $a_1 \in \mathcal{A}^\circ$, $a_2 \in \mathcal{A}^\alpha$, $a_1, a_2 \in A_p(f)$



On récapitule la construction des arcs. Les couleurs des différents types d'arcs sont celles utilisées dans le tableau précédent.

Le lemme suivant est une conséquence immédiate de cette construction :

Lemme IV.2 *$\mathcal{C}$ est une coupe sur $\mathcal{G}$ si, et seulement, si la configuration $f^{\mathcal{C}}$ correspondante est un α-*expansion move de f°.

Lemme IV.3 *Le coût de la coupe $\mathcal{C}$ est fini si, et seulement si, la configuration $f^{\mathcal{C}}$ correspondante est unique.*

Théorème IV.2 *Soit $\mathcal{C}$ la coupe minimale de G. Alors $f^{\mathcal{C}}$ est le α-expansion move de f°, qui minimise l'énergie E.*

On se reportera à l'annexe pour une démonstration complète des différents résultats.

Algorithme IV.2 : MINIMISATION D'UNE ÉNERGIE PAR GRAPH CUT

initialize Soit f° une configuration
for all $\alpha \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$
do {
construire le graphe correspondant
appliquer l'algorithme de Max-Flow/Min-Cut
déterminer le poids $\text{cap}(\mathcal{C})$ de la coupe minimale $\mathcal{C}$ obtenue
if $\text{cap}(\mathcal{C}) + C < E(f^\circ)$
then $f^\circ := f$
}
return (f°)

IV.6 Choix des paramètres

La méthode présentée ci-dessus utilise plusieurs paramètres, qui correspondent aux pénalités affectées à chacun des termes de l'énergie. Le code étudié a besoin de deux paramètres : `lambda` et `K`, qui correspondent respectivement à la pénalité liée à la régularité et celle liée à l'occlusion. Il est possible de les entrer manuellement, mais le code permet également de les calculer automatiquement.

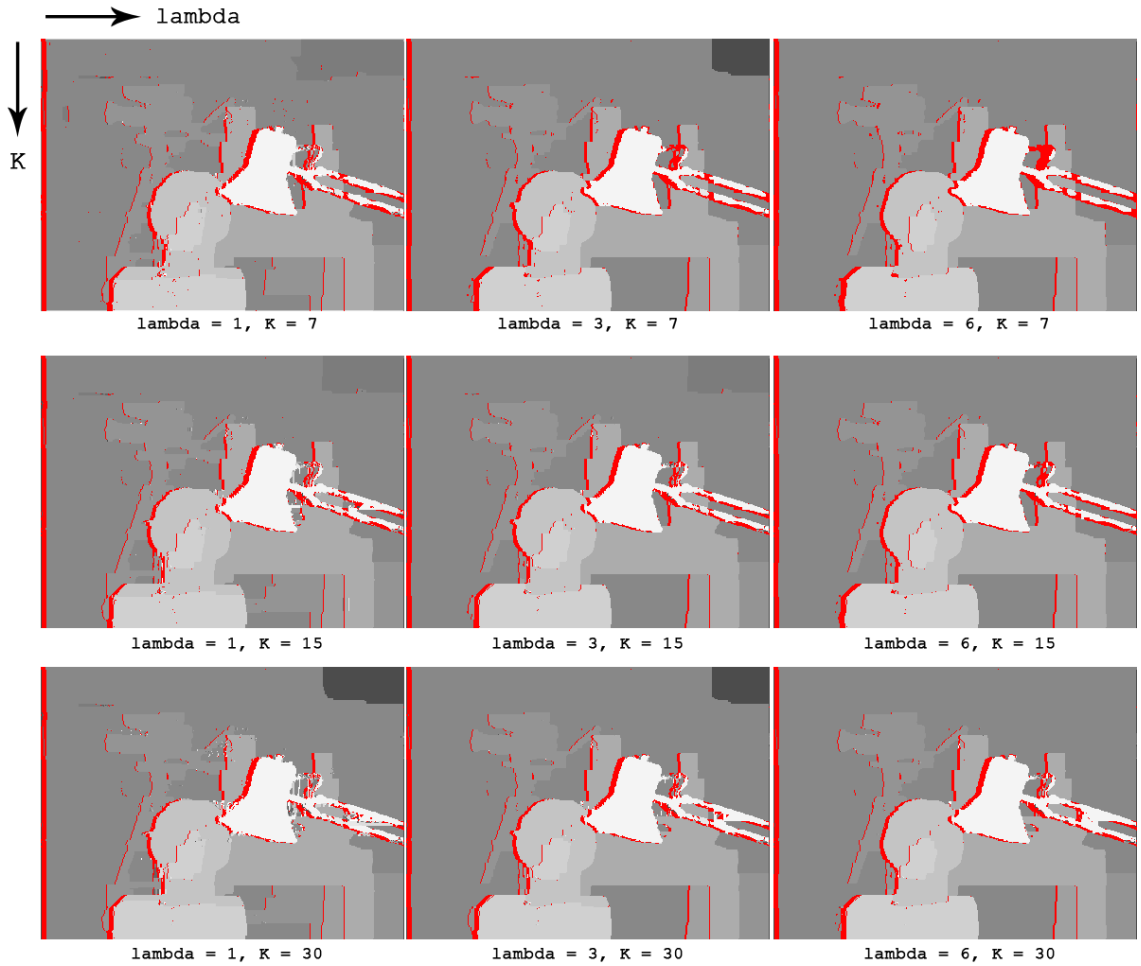


FIGURE 19 – Résultats avec différentes valeurs de `K` et de `lambda`. Lorsque `lambda` augmente, la régularité aussi ; on peut le constater au niveau des branches de la lampe. (Résultats obtenus sous Linux)

Pour chacun de ces paramètres, il existe [15] un intervalle dans lequel les résultats ne varient que sensiblement ; ce n'est que hors de ces intervalles que les résultats commencent à se dégrader, étant soit trop lisses, soit au contraire trop bruités. Cependant, il semble que ces intervalles diffèrent énormément selon les paires d'images. KOLMOGOROV propose une heuristique qui permet de choisir automatiquement les paramètres, censée éviter les deux écueils susmentionnés.

Pour ce faire, considérons le terme de données $E_{\text{données}}$:

$$E_{\text{données}}(f) = \sum_{a \in \mathcal{A}} D_a(f(a)) = \sum_{a \in \mathcal{A}} (D_a(f(a)) - K) + K\,B$$

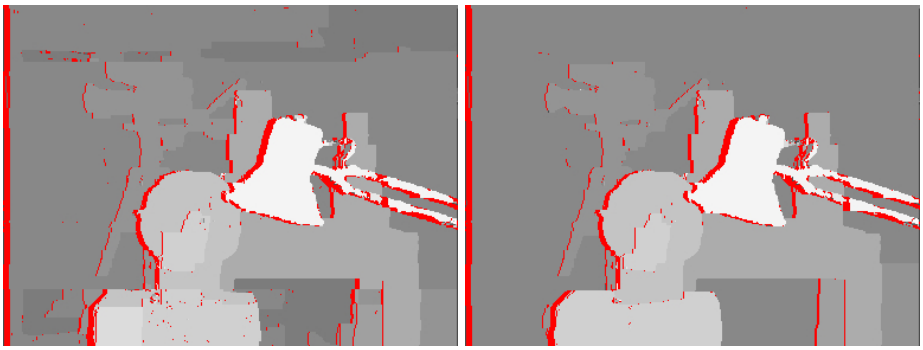
avec

$$D_a(f(a)) - K = \begin{cases} D(a) - K & \text{si } f(a) = 1 \\ 0 & \text{sinon} \end{cases}$$

Le terme $K\,B$ est constant, donc la pénalité est soit nulle s'il y a occlusion, soit vaut $D(a) - K$. De là, on constate que si K est trop grand, les pénalités liées à la fidélité seront négatives, et donc la minimisation aura tendance à associer des pixels, en autorisant des correspondances entre pixels de différences d'intensité importante. Si K est au contraire trop faible, la pénalité sera positive, donc les occlusions seront favorisées.

Pour choisir un K convenable, on décide de prendre la plus petite valeur qui discrimine les 25 % meilleures mises en correspondance pour un pixel p. Pour cela, on commence par calculer, pour tout pixel p, $D(a)$ avec a un assignement impliquant le pixel p. On les ordonne et on pose $D(p)$ la k-ème valeur (en prenant la valeur entière), avec k égal à $0,25$ fois le nombre de niveaux de profondeurs dans l'image ($x_{\max} - x_{\min}$ en général, si on est en géométrie épipolaire et pas au bord), ou 3 si ce nombre est inférieur à 3. Supposons en effet qu'on ait 20 niveaux de profondeur dans l'image ; k vaut donc 4. Soit p un pixel. Il y a k assignements possibles impliquant p, donc pour seulement 25 % d'entre eux, $D(a) - D(p)$ aura une contribution négative. On choisit alors K le nombre moyen des $D(p)$.
Pour fixer la valeur de λ, on part de l'intuition que les pénalité de régularité et d'occlusion doivent être équilibrées. [16] choisit de les prendre l'un proportionnel à l'autre, avec la relation $\lambda = K/5$.

Dans le cadre de la mise en ligne du code sur le site IPOL [2], nous avons dû, pour des raisons techniques, faire tourner l'algorithme en coupant au préalable les images en six bandes horizontales ; le code a alors calculé les paramètres automatiquement pour chacune des bandes, qui se sont alors trouvées de valeur parfois très différentes.



(a) Paramètres libres sur chaque bande (b) Paramètre unique

Les résultats montrent ainsi clairement la dépendance des paramètres aux données. Ceci joue en la défaveur de la méthode, car la détermination d'un relief doit être *a priori* stable par changement de fenêtre verticale dans le cas d'un décalage horizontale de la caméra.

Pour obtenir un résultat plus probant, il nous a donc fallu modifier le code afin de faire calculer les paramètres automatiques *avant* de couper l'image et de lancer le programme avec les paramètres ainsi obtenus. Plus précisément, on a fait tourner le code sur des bandes en ajoutant des pixels en haut et en bas, car la fonctionnelle d'énergie prend en compte les voisinages des pixels.

FIGURE 20 – Pour l'utilisation du code en ligne, on a découpé les images en six bandes horizontales.

IV.7 Mesure de dissimilarité de Birchfield et Tomasi

On rappelle que le terme de fidélité de l'énergie fonctionnelle est donné par :

$$E_{\text{fidélité}} = \sum_{a \in A(f)} D(a)$$

où D mesure la différence d'intensité entre deux pixels mise en correspondance. Cette fonction est classiquement choisie comme étant la norme L^1 ou la norme L^2 euclidienne, ou encore la distance quadratique. Cependant, ce genre de norme peut pénaliser des assignements pourtant corrects. En effet, si l'intensité dans une scène est continue par morceaux, les pixels des deux images que l'on en a sont des échantillons discrets. Si l'intensité autour d'un point de la scène n'est pas constante, l'intensité des deux pixels homologues correspondant à ce point peut différer. Ainsi, on pénalise bien que la correspondance ait été bien faite. Il faut donc trouver une mesure qui reste robuste par échantillonnage, c'est-à-dire qui donnent les mêmes résultats quel que soit l'échantillonnage.

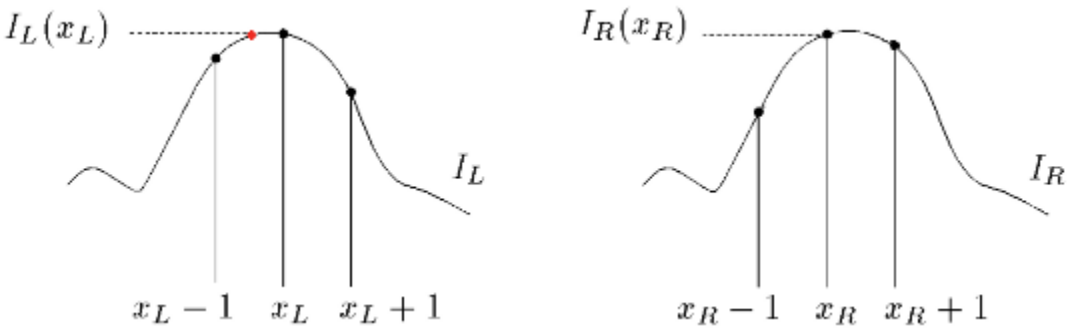


FIGURE 21 – x_L et x_R sont homologues ; l'échantillonnage fait que les deux pixels ne correspondent pas exactement au même point (le pixel x_R est représenté en rouge) et n'ont donc pas exactement la même intensité, bien qu'il s'agisse de la meilleure mise en correspondance possible.

On va décrire ici une mesure qui est assez robuste, donnée dans [4, 3]. Ce n'est pas la meilleure dans cette catégorie, mais elle a l'avantage, ainsi qu'on va le voir, d'être extrêmement simple à calculer. On se place dans le cadre de la géométrie épipolaire. On considère une ligne de la scène. On note i_L et i_R les deux fonctions continues (par morceaux) associées à l'intensité des points de la ligne dans chacune des deux images ; I_R et I_L sont alors les équivalents échantillonnés, qui sont donc deux tableaux de valeurs d'intensité.

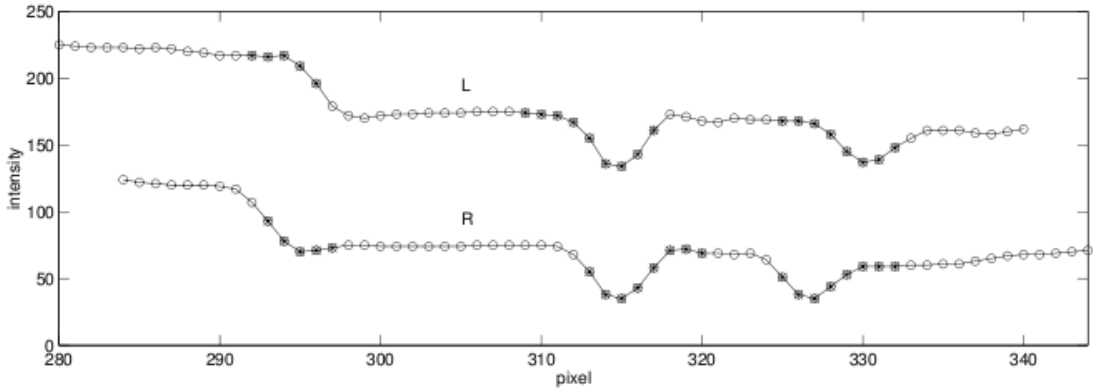


FIGURE 22 – Les deux lignes continues correspondent à i_L et i_R, décalées afin de voir les correspondances. Les ronds symbolisent l'échantillonnage des deux images. (Image : [3])

On pose par ailleurs $\hat{I}_L$ (resp. $\hat{I}_R$) les interpolations linéaires de I_L (resp. I_R). Soit x_L un pixel de l'image de gauche et x_R son homologue dans l'image de droite. On définit la quantité suivante

$$\bar{D}_{BT}(x_L, x_R, I_L, I_R) = \min_{x_R - \frac{1}{2} \leq x \leq x_R + \frac{1}{2}} |I_L(x_L) - \hat{I}_R(x)|$$

et la quantité symétrique

$$\bar{D}_{BT}(x_R, x_L, I_R, I_L) = \min_{x_L - \frac{1}{2} < x < x_L + \frac{1}{2}} |\hat{I}_L(x) - I_R(x_R)|$$

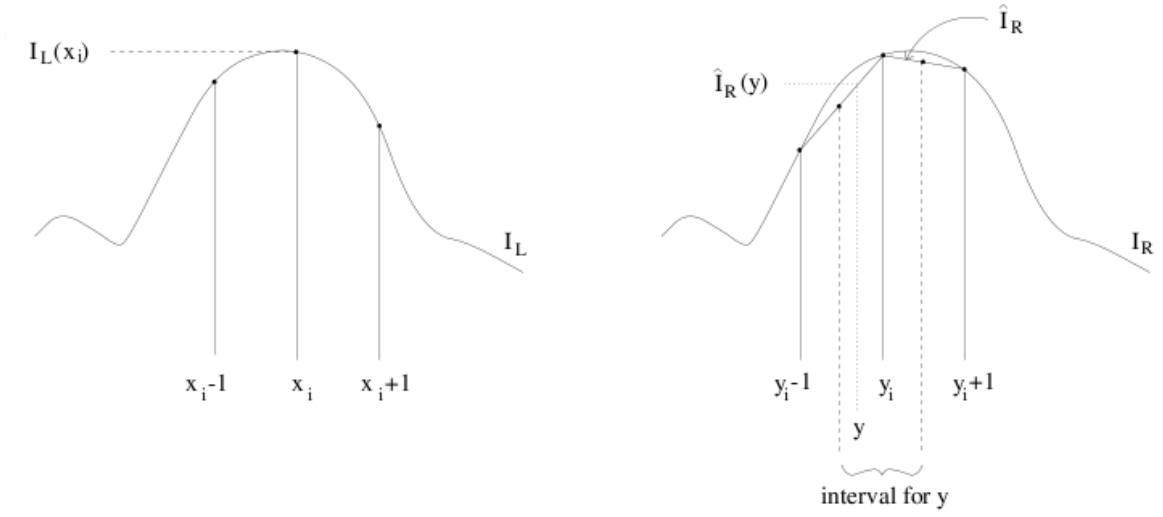


FIGURE 23 – Définition de $\bar{D}_{BT}(x_L, x_R, I_L, I_R)$. (Image : [3])

On définit alors la *dissimilarité* d entre les deux pixels x_L et x_R de manière symétrique comme étant le minimum des deux quantités précédentes

$$D_{BT}(x_L, x_R) = \min\{\bar{D}_{BT}(x_L, x_R, I_L, I_R), \bar{D}_{BT}(x_R, x_L, I_R, I_L)\}$$

On cherche à exprimer plus simplement D_{BT}. Pour cela, on remarque que les extrema (locaux) d'une fonction affine par morceaux sont les points de discontinuité de la pente, c'est-à-dire ici les points $x_R - \frac{1}{2}$,

x_R ou $x_R + \frac{1}{2}$ (pour l'image droite par exemple). On commence par poser

$$I_R^- := \hat{I}_R\left(x_R - \frac{1}{2}\right) \quad \text{et} \quad I_R^+ := \hat{I}_R\left(x_R + \frac{1}{2}\right)$$

que l'on peut écrire

$$I_R^- = \frac{1}{2}(I_R(x_R) + I_R(x_R - 1)) \quad \text{et} \quad I_R^+ = \frac{1}{2}(I_R(x_R) + I_R(x_R + 1))$$

On pose alors $I_{\min} := \min\{I_R^-, I_R^+, I_R(x_R)\}$ et $I_{\max} := \min\{I_R^-, I_R^+, I_R(x_R)\}$. Ces quantités ainsi définies, on a

$$\bar{D}_{BT}(x_R, x_L, I_R, I_L) = \max\{0, I_L(x_L) - I_{\max}, I_{\min} - I_L(x_L)\}$$

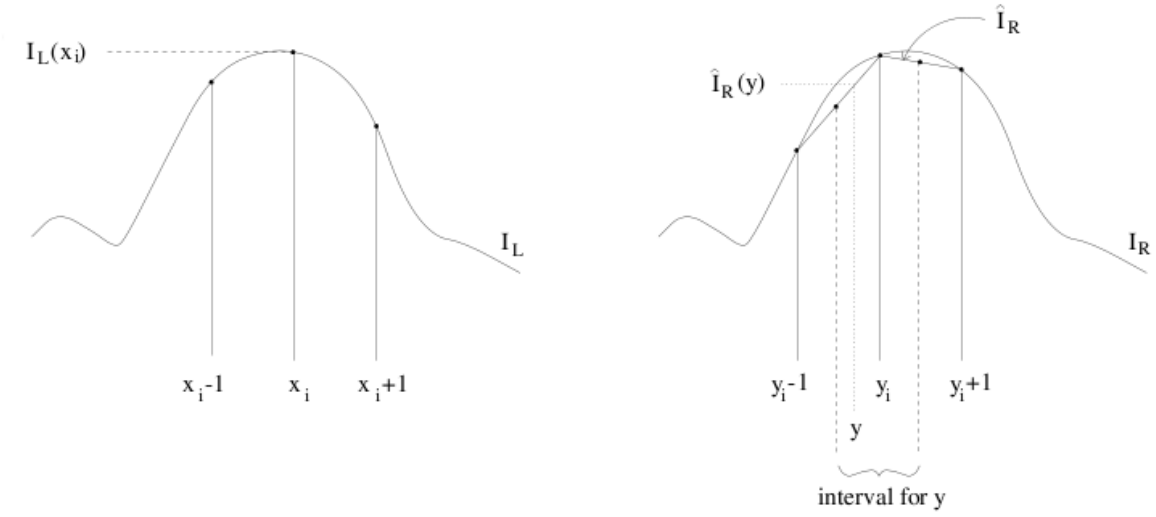
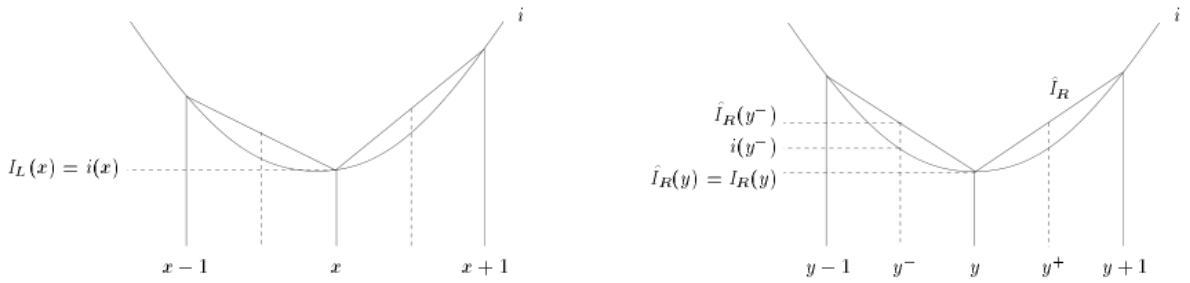


FIGURE 24 – Calcul de $\bar{D}_{BT}(x_L, x_R, I_L, I_R)$. (Image : [3])

Montrons que cette mesure est bien (suffisamment) invariante par échantillonnage. On se place dans le cas où les images sont prises sans distortion aucune. Alors, si x_L et x_R correspondent, l'intensité i_L au voisinage de x_L et i_R au voisinage de x_R sont égales car la scène (et donc l'intensité) est continue par morceaux. On note $i := i_L = i_R(\cdot + d)$ la fonction d'intensité commune (à un décalage près) aux deux images sur un tel voisinage. Dans ce cas-là, si x_L et x_R correspondent au même point, d'intensité respective $i(x_L)$ et $i(x_R + d)$. Du fait de l'enchantillonnage, $|x_L - x_R| \leq \frac{1}{2}$. Il faut donc montrer que, dans ces conditions, $D_{BT}(x_L, x_R) = 0$.
On commence par montrer ce résultat dans un cas plus faible où l'intensité est concave ou convexe sur l'intervalle sur lequel on la considère :

Théorème IV.3 *On suppose que i est convexe ou concave sur un intervalle A et que x_L et x_R tels que $\left[x_L - \frac{1}{2}, x_L + \frac{1}{2}\right] \subset A$ et $\left[x_R - \frac{1}{2}, x_R + \frac{1}{2}\right] \subset A$. Si $|x_L - x_R| \leq \frac{1}{2}$, alors $D_{BT}(x_L, x_R) = 0$.*

Preuve : Pour montrer ce résultat, dans le cas où la fonction est convexe par exemple, il faut remarquer que si $x \leq y \leq z$, alors $i(y) \leq i(x)$ ou $i(y) \leq i(z)$ (on le démontre par l'absurde). On prouve alors que $\bar{D}_{BT}(x, y, I_L, I_R) = 0$ (ou $\bar{D}_{BT}(y, x, I_L, I_R) = 0$, car la quantité $D_{BT}(x_L, x_R)$ est positive). En posant $y^- := y - \frac{1}{2}$ et $y^+ := y + \frac{1}{2}$ (cf. figure suivante), la remarque précédente nous assure que $|x - y| \leq \frac{1}{2}$ implique que $i(x) \leq i(y^-)$ ou $i(x) \leq i(y^+)$.



On se place dans le premier cas. Alors, si $I_R(y) \leq I_L(x)$ (le cas contraire prouverait que $\bar{D}_{BT}(y, x, I_L, I_R) = 0$), alors on a

$$\hat{I}_R(y) = I_R(y) \leq I_L(x) = i(x) \leq i(y^-)$$

et comme i est convexe, $i(y^-) \leq \hat{I}_R(y^-)$. Ainsi,

$$\hat{I}_R(y) \leq I_L(x) \leq \hat{I}_R(y^-)$$

Puisque $\hat{I}_R$ est continue sur l'intervalle $[y^-, y]$, le théorème des valeurs intermédiaires assure qu'il existe un $y' \in [y^-, y]$ tel que $\hat{I}_R(y') = I_L(x)$ et dans ce cas, en utilisant la définition de $\bar{D}_{BT}(x, y, I_L, I_R)$, $\bar{D}_{BT}(x, y, I_L, I_R) = 0$. ■

Dans le cas d'un point d'inflexion, en pratique, l'intensité se comporte comme une fonction linéaire, pourvu que l'intensité varie continûment.

Théorème IV.4 *Si i est une fonction linéaire non constante sur un intervalle A tel que* $\left[x_L - \frac{1}{2}, x_L + \frac{1}{2}\right] \subset A$ *et* $\left[x_R - \frac{1}{2}, x_R + \frac{1}{2}\right] \subset A$, *alors $D_{BT}(x_L, x_R) = 0$ si, et seulement si, $|x_L - x_R| \leq \frac{1}{2}$.*

Ce résultat nous assure, par ailleurs, que dans ce cas, une mise en correspondance fausse est toujours pénalisée, si cette fonction linéaire est de pente non nulle.

On vient donc de montrer que pour des régions (voisinages de point) où l'intensité était convexe, concave, ou linéaire, la mesure que l'on a construite était bien robuste à l'échantillonnage.

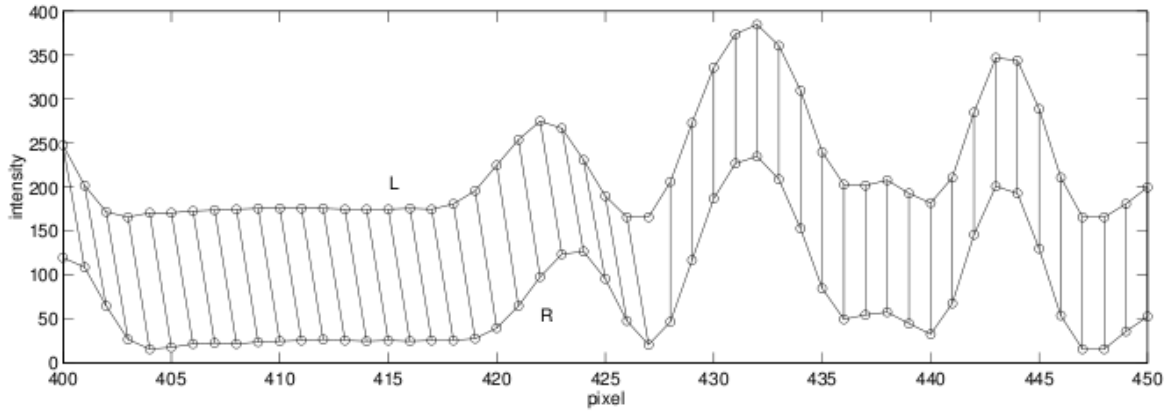


FIGURE 25 – Exemple de mise en correspondance. (Image : [3])

La figure ci-dessus donne un exemple de mise en correspondance entre une ligne de l'image de gauche (haut) et une ligne de l'image de droite (bas), dépendant d'un certain échantillonnage donné. Si l'on

calcule pour chaque assignement actif la pénalité de fidélité en utilisant comme mesure la valeur absolue de la différence (SAD), on obtiendrait :

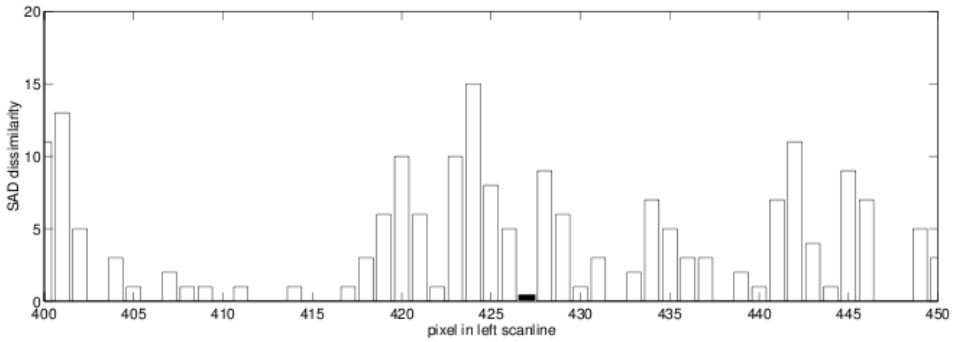


FIGURE 26 – Résultat avec SAD. (Image : [3])

Il est visible que dans cet exemple précis, la mise en correspondance est la meilleure que l'on puisse faire. Cependant, on voit ici que la pénalité du coût de fidélité est parfois très grande, et on peut donc raisonnablement se demander si une autre configuration, moins bonne, ne pourrait pas toutefois donner un terme de fidélité plus petit. On peut comparer ce résultat avec celui obtenu avec la mesure de BIRCHFIELD–TOMASI :

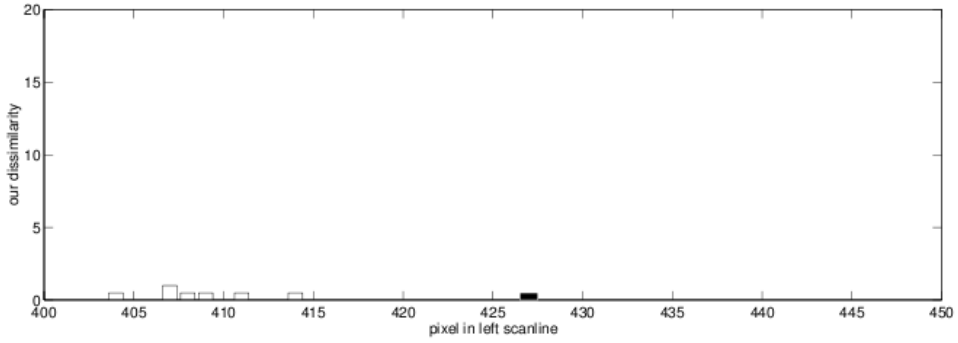


FIGURE 27 – Résultat avec BT. (Image : [3])

Le résultat est bien celui attendu, si ce n'est pour les premiers pixels (en partant de la gauche), mais ceci peut s'expliquer par le fait que la pente de l'intensité est quasi-nulle ; il est difficile de voir à l'œil nu si les intensités sont vraiment égales sur cette intervalle. Il est donc possible que la mise en correspondance introduise réellement un terme de fidélité non nul ici.

La mesure de BIRCHFIELD–TOMASI est donc très robuste à échantillonnage dans les régions où l'intensité est continue, sous réserve que les deux images ne sont ni déformées, ni bruitées. Ceci en fait une mesure plus intéressante que celles classiquement utilisée en stéréo, qui dépendent beaucoup trop de l'échantillonnage. Dans la méthode de KOLMOGOROV–ZABIH [15], le terme de données est une variante bi-dimensionnelle de cette mesure, calculée horizontalement et verticalement, et dont on prend la moyenne : si $p = (p_x, p_y)$ et $q = (q_x, q_y)$ sont homologues, alors on calcule :

$$\begin{cases} I^-_{R_x} = \dfrac{1}{2}(I_R(q_x, q_y) + I_R(q_x - 1, q_y)) \\ I^+_{R_x} = \dfrac{1}{2}(I_R(q_x, q_y) + I_R(q_x + 1, q_y)) \end{cases} \quad \text{et} \quad \begin{cases} I^-_{R_y} = \dfrac{1}{2}(I_R(q_x, q_y) + I_R(q_x, q_y - 1)) \\ I^+_{R_y} = \dfrac{1}{2}(I_R(q_x, q_y) + I_R(q_x, q_y + 1)) \end{cases}$$

ce qui nous donne

$$\begin{cases} I_{\min_x} = \min\{I^-_{R_x}, I^+_{R_x}, I_{R_x}(q_x, q_y)\} \\ I_{\max_x} = \min\{I^-_{R_x}, I^+_{R_x}, I_{R_x}(q_x, q_y)\} \end{cases} \quad \text{et} \quad \begin{cases} I_{\min_y} = \min\{I^-_{R_y}, I^+_{R_y}, I_{R_y}(q_x, q_y)\} \\ I_{\max_y} = \min\{I^-_{R_y}, I^+_{R_y}, I_{R_y}(q_x, q_y)\} \end{cases}$$

On peut alors calculer

$$\begin{cases} \bar{D}_{BT_x}(p, q, I_R, I_L) = \max\{0, I_L(p) - I_{\max_x}, I_{\min_x} - I_L(p)\} \\ \bar{D}_{BT_y}(p, q, I_R, I_L) = \max\{0, I_L(p) - I_{\max_y}, I_{\min_y} - I_L(p)\} \end{cases}$$

et symétriquement $\bar{D}_{BT_x}(q, p, I_L, I_R)$ et $\bar{D}_{BT_x}(q, p, I_L, I_R)$, pour finalement obtenir les valeurs $D_{BT_x}(p, q)$ et $D_{BT_y}(p, q)$ comme étant les minimums des deux termes ci-dessus correspondants :

$$\begin{cases} D_{BT_x}(p, q) = \min\{\bar{D}_{BT_x}(p, q, I_R, I_L), \bar{D}_{BT_x}(q, p, I_L, I_R)\} \\ D_{BT_y}(p, q) = \min\{\bar{D}_{BT_y}(p, q, I_R, I_L), \bar{D}_{BT_x}(q, p, I_L, I_R)\} \end{cases}$$

On pose alors

$$D(\langle p, q \rangle) = \frac{1}{2}\left(D_{BT_x}(p, q) + D_{BT_y}(p, q)\right)$$

C'est cette mesure qui est utilisée dans le terme de fidélité.

V Résultats expérimentaux

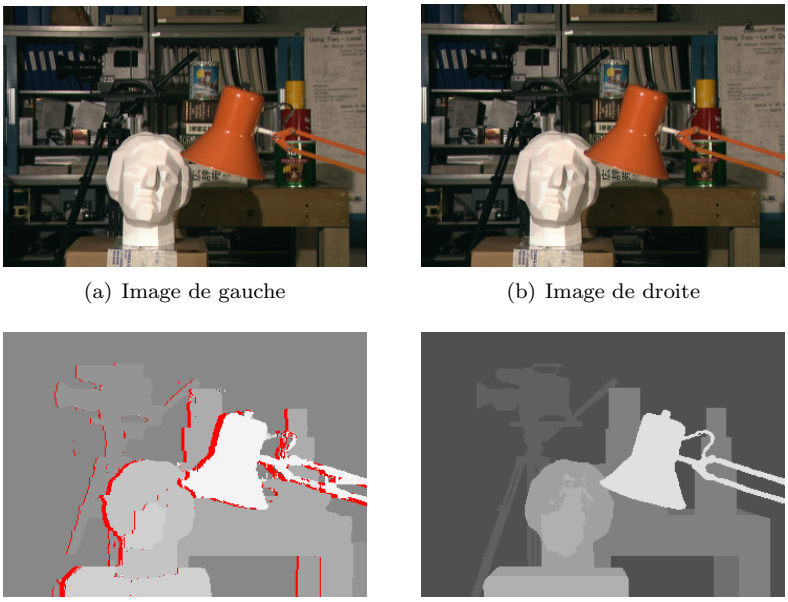
Nous présentons dans cette section les résultats que nous avons obtenus en faisant tourner le code de KOLMOGOROV et ZABIH sur des paires d'images dont on connaissait parfois la disparité, ou que nous avons dû estimer grossièrement nous-mêmes sinon.

V.1 Images Middlebury

La banque de données Middlebury contient des images assez typiques, présentant des régions souvent très texturées, des scènes comportant beaucoup de surfaces planes, et avec un rapport baseline sur profondeur (b/h) relativement constant. Cela peut biaiser les résultats car certains paramètres voire algorithmes peuvent être optimisés pour ce genre de scène.

FIGURE 28 – Exemples d'images du *benchmark* Middlebury. Les trois premières scènes semblent très artificielles, car il faut créer des surfaces suffisamment texturées pour que les algorithmes soient efficaces.

Tsukuba Cette scène est une des images de référence en stéréovision. Elle fait partie de celles sur lesquelles les résultats sont les plus impressionnants. Les différents niveaux du relief sont bien rendus, que ce soit au niveau du buste central, de la lampe, ou même de la caméra au fond. Les occlusions sont bien détectées (ce qui est une des forces de cette méthode). Cela est certainement dû au fait que l'image est bien texturée (en particulier en ce qui concerne le fond de la scène), ce qui en fait une paire "facile".



(a) Image de gauche

(b) Image de droite

(c) Résultats (`lambda : 3 (AUTO)`, `K : 15 (AUTO)`)

(d) Vérité-terrain

FIGURE 29 – Tsukuba

Le tableau suivant compare quantitativement ce résultat avec d'autres méthodes (chiffres : [15]).

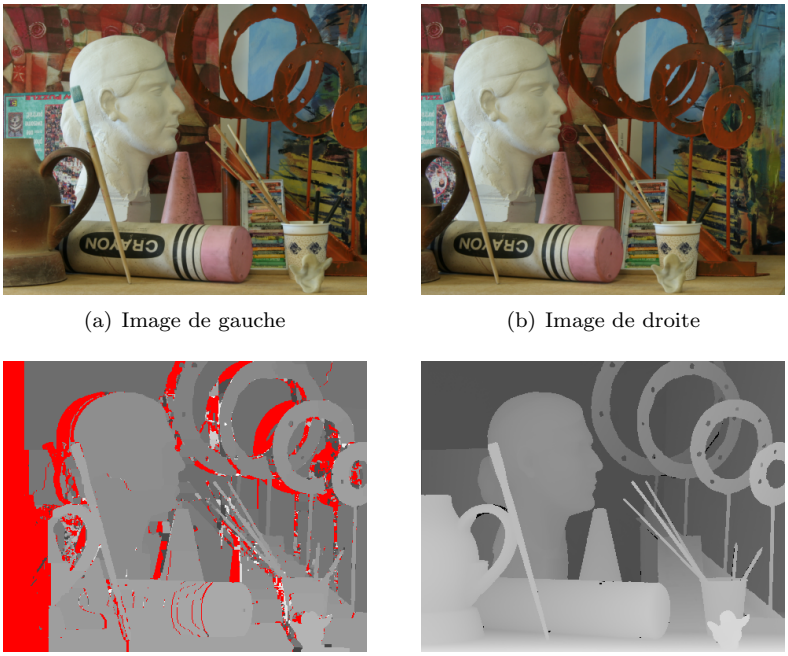
Méthode	Disparités		Occlusions	
	Erreurs	Grosses erreurs	Faux négatifs	Faux positifs
KOLMOGOROV et ZABIH [16] (Graph Cut avec *expansion moves*)	6,7 %	1,9 %	42,6 %	1,1 %
BOYKOV, VEKSLER et ZABIH [8] (Graph Cut avec *swap moves*)	6,7 %	2,0 %	82,8 %	0,3 %
ZITNICK et KANADE [24] (basé sur une condition d'unicité et de continuité)	12,0 %	2,6 %	87,3 %	0,8 %
Corrélation	28,5 %	1,9 %	87,3 %	6,1 %

La méthode [8] utilise non seulement les α-*expansion moves*, mais également les $\alpha\,\beta$-*swap moves*, qui consistent à permuter des disparités de valeurs α et β, pour faire diminuer l'énergie. Une telle minimisation se fait également par Graph Cuts. On constate enfin sans surprise que la méthode par corrélation gère très mal les occlusions. Les deux premières colonnes ne concernent que les pixels non occlus de la scène ; dans ce cas, un pixel déclaré occlus est considéré comme une grosse erreur. Une erreur est une disparité incorrecte, tandis qu'une grosse erreur est une disparité qui diffère de plus de 1 de la disparité réelle.

Cependant, il nous faut faire une remarque concernant ce résultat : en faisant tourner le code sous Linux, nous avons obtenu un résultat moins satisfaisant (une disparité fausse dans le coin supérieur droit de l'image). Il s'avère que dans le code écrit par KOLMOGOROV, certaines fonctions sont écrites différemment selon le système d'exploitation (Windows ou Linux) ; il semblerait donc que les équivalences ont été mal gérées. Cela ne semble pas concerner la détermination des paramètres, car nous avons entré dans les expériences ci-dessus les paramètres manuellement. Les résultats que nous présenterons seront donc, sauf mention contraire, obtenus en faisant tourner le code sous Windows.

Art Cette scène du *benchmark* est intéressante en raison des nombreux éléments fins qu'il comporte (pinceaux et tiges), qui sont en général mal gérés par d'autres méthodes (par exemple, les méthodes

locales peinent à estimer correctement la disparité aux bords des objets, car dans la fenêtre considérée, la scène peut avoir beaucoup changé d'une image à l'autre). Ici, l'algorithme s'en sort visiblement très bien pour ces éléments particuliers de la scène. Il faut toutefois noter la présence de beaucoup de déchets, en particulier pour le fond de la scène (les zones blanches à la place des occlusions, alors que la profondeur est censée être grande).



(a) Image de gauche

(b) Image de droite

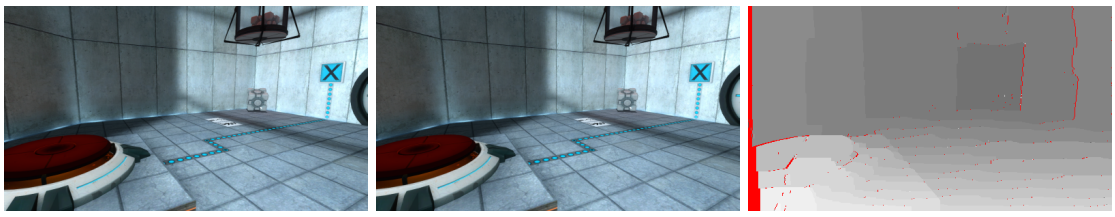
(c) Résultats (`lambda : 33 (AUTO)`, `K : 165 (AUTO)`)

(d) Vérité-terrain

FIGURE 30 – Art

V.2 Autres images

Portal La scène a été extraite d'un jeu vidéo [6] et représente une salle, avec au premier plan un bouton géant.



(a) Image de gauche

(b) Image de droite

(c) Résultats (`lambda : 27 (AUTO)`, `K : 135 (AUTO)`)

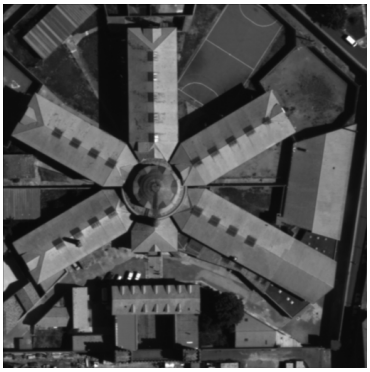
FIGURE 31 – Portal

Les résultats de l'algorithme pour cette paire (avec les paramètres par défaut) sont relativement bon, en ce sens que le relief du bouton est bien rendu, ainsi que la profondeur de la scène (on observe bien les dégradés auxquels on s'attendait). Cet exemple montre cependant à quel point le caractère quantifié de la disparité produite peut devenir une faiblesse, puisqu'ici, le coin de la pièce disparaît au profit d'une

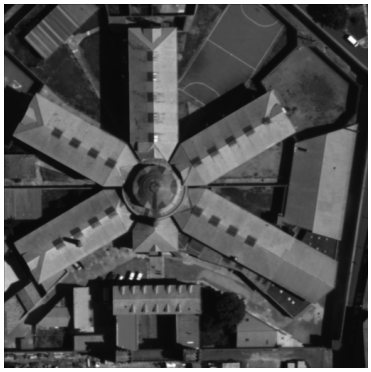
6. *Portal*, développé par Valve Software, 2007

succession de plans parallèles, qui ne reflètent donc que très grossièrement la réalité. On voit donc ici à quel point une quantification de la disparité peut parfois être inadaptée pour certains types de scènes.

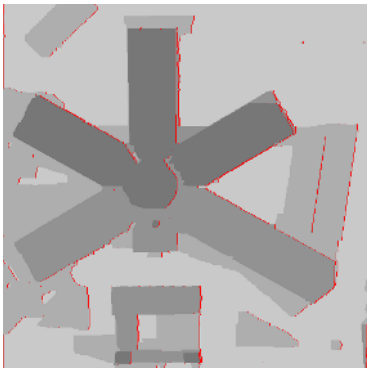
Prison On retrouve ce même travers dans cette scène, qui est une vue aériennne d'une prison à Toulouse. La disparité est ici très faible (de -15 à 15 pixels sur 512), et ce genre d'images est assez caractéristique de l'utilisation qui pourrait être faite des algorithmes de stéréovision, à savoir la reconstruction du relief par images aériennes (voire satellitaires).



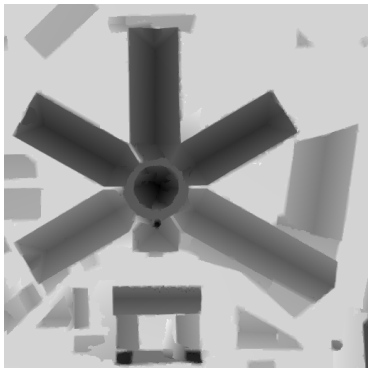
(a) Image de gauche



(b) Image de droite



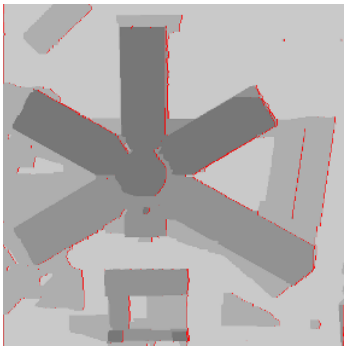
(c) Résultats (`lambda : 7 (AUTO)`, `K : 35 (AUTO)`)

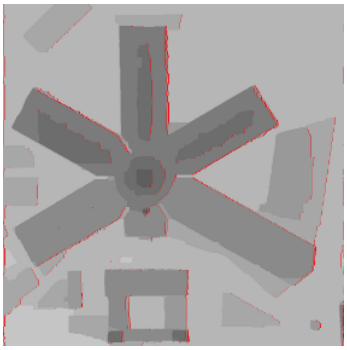


(d) Vérité-terrain

FIGURE 32 – Prison

Si le résultat reste bon, on a toutefois perdu pas mal d'informations, notamment aux niveaux du toit, qui paraît plat alors que ce n'est pas le cas. Pour obtenir une précision supplémentaire, on a fait tourner l'algorithme sur ces mêmes images, mais zoomées deux fois (à droite).

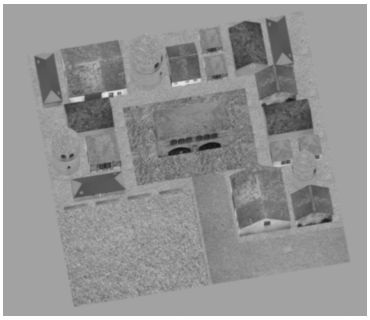




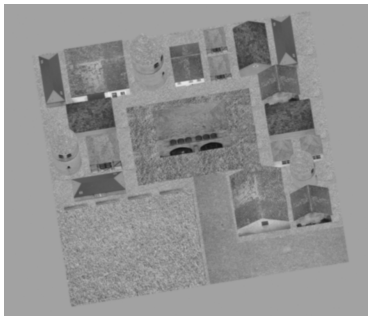
On peut ainsi obtenir davantage de relief au niveau du dôme et d'une partie du toit, mais cela se fait au prix fort : on a dû faire tourner le code sur des images quatre fois plus grosses, ce qui double le rang

de disparité (c'est-à-dire l'intervalle dans lequel se situe la disparité), et donc le nombre d'assignements. Chaque graphe est donc plus de quatre fois plus grand, et comme il y a deux fois plus de valeurs de disparités possibles, chaque itération comporte donc quatre fois plus d'*expansion moves*. La complexité de cet algorithme n'est donc pas linéaire (alors qu'il ne faut pas plus de cinq minutes pour obtenir un résultat avec les images originales (512×512), le faire sur des images zoomées (1024×1024) demande plus d'une demi-heure.

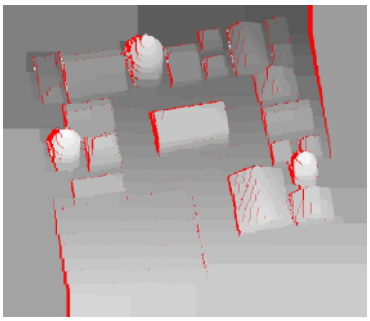
Village Cette paire-ci représente la vue en plongée d'un ensemble de maisons. Le résultat est globalement assez satisfaisant, mais est totalement faux pour les régions noires des images d'origine. Cela est dû au fait que ce fond est uni, et même notre œil est incapable de décider quelle est la profondeur de ces régions. Ceci constitue une des plus grandes difficultés de la stéréovision, car il est difficile (voire impossible) d'attribuer une disparité à une région sans information (comprendre sans motif ou texture). Comme toutes les configurations se valent dans ce cas-là, l'algorithme se trompe généralement, mais produit la disparité la moins coûteuse, donc généralement assez lisse.



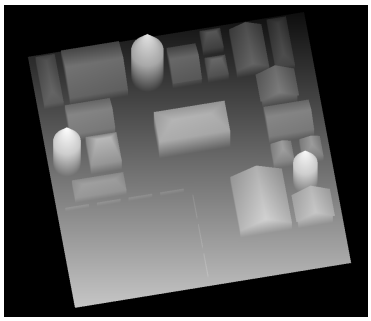
(a) Image de gauche



(b) Image de droite



(c) Résultats (`lambda : 7 (AUTO)`, `K : 35 (AUTO)`)



(d) Vérité-terrain

FIGURE 33 – Village

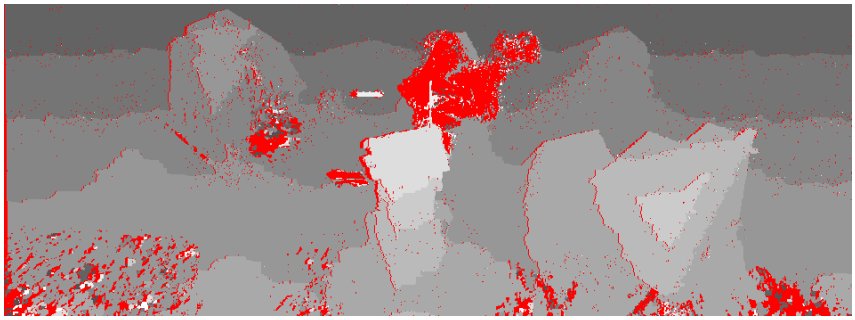
Plage Cette scène est également tirée d'un jeu vidéo ; il représente une plage, sur laquelle se dressent trois constructions. Si l'algorithme donne des résultats satisfaisants pour cette scène (on pourra par exemple regarder la tour centrale), ce sont les endroits où il a échoué qui rendent cet exemple intéressant, l'échec étant localisé aux zones rouges.
On peut distinguer :

- la zone au-dessus de la tour centrale : elle correspond à la présence du drapeau, qui bouge (à cause du vent), car les deux prises n'étaient pas simultanées (situations qui peuvent arriver, surtout lors de prises de vue par satellite) ; par ailleurs, on peut remarquer la présence de fumée, qui rend plus difficile la mise en correspondance ;
- la fontaine de gauche et le bas de l'image : ces deux zones correspondent à la présence d'eau, qui bouge (l'eau de la fontaine s'écoule et il y a des vagues dans la mer), et qui, surtout, présente des

reflets (ce qui est l'une des plus grandes difficultés rencontrées en stéréo).
Cette paire d'images est donc assez typique des difficultés inhérentes à la stéréoscopie, dès que les scènes sont moins artificielles et les conditions moins favorables.

(a) Image de gauche

(b) Image de droite



(c) Résultats (`lambda : 14 (AUTO)`, `K : 70 (AUTO)`)

FIGURE 34 – Plage

Conclusion

La recherche en stéréovision est très active, et des méthodes sont régulièrement proposées par les chercheurs. Grâce à la création du *benchmark* Middlebury, les plus efficaces d'entre elles obtiennent plus de visibilité, ce qui crée une véritable émulation dans la communauté. Cependant, bien que certains résultats soient impressionnants, il reste de très nombreux progrès à faire en la matière, comme on a pu le constater avec l'algorithme de KOLMOGOROV et ZABIH, qui fait pourtant partie des méthodes les plus efficaces. Un des enjeux majeurs de ce genre de méthodes est l'élimination des paramètres (qui empêchent l'utilisation automatique des algorithme et qui, par nature, posent le problème de l'unicité de la solution). L'intégration de processus complémentaires constitue par ailleurs une voie intéressante à explorer, les méthodes utilisant le *Belief Propagation* par exemple étant les mieux classées sur le banc de test Middlebury.

Des méthodes toujours plus efficaces impliquent malheureusement une complexité toujours croissante ; ainsi, les méthodes basées sur des calculs de flot maximal sont-elles souvent limitées par la taille des graphes, de l'ordre de celles des images, ce qui les rend peu adaptées à une utilisation pour l'imagerie satellitaire en particulier, où les images ont des tailles beaucoup trop importantes. Cependant, on peut citer [21] qui propose de couper les graphes en plusieurs parties pour calculer séparément les flots dans chacune des parties, permettant ainsi les calculs parallèles. Cela devrait augmenter la rapidité des calculs et ainsi rendre les méthodes plus intéressantes.

Références bibliographiques

[1] Middlebury Stereo Vision Page : `http://vision.middlebury.edu/stereo`.

[2] IPOL : Image Processing On Line : `http://www.ipol.im`.

[3] Stan Birchfield and Carlo Tomasi. Depth discontinuities by pixel-to-pixel stereo. Technical report, Stanford University, 1996.

[4] Stan Birchfield and Carlo Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, April 1998.

[5] J.A. Bondy and U.S.R. Murty. *Graph Theory*. Springer, 2008.

[6] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, September 2004.

[7] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *Conference on Computer Vision and Pattern Recognition*, 1998.

[8] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, November 2001.

[9] Myron Z. Brown, Darius Burschka, and Gregory D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25, August 2003.

[10] Vicent Caselles and Nicolas Papadakis. Multi-label depth estimation for graph cuts stereo problems. Technical report, 2009.

[11] Nicos Christofides. *Graph Theory – an Algorithmic Approach*. Academic Press, 1975.

[12] Dorin Comaniciu and Peter Meer. Mean shift : A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, May 2002.

[13] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[14] Andreas Klaus, Mario Sormann, and Konrad Karner. Segment-based stereo matching using belief propagation and a selt-adapting dissimilary measure. In *International Conference on Pattern Recognition*, 2006.

[15] Vladimir Kolmogorov. *Graph Based Algorithms for Scene Reconstruction from Two or More Views*. PhD thesis, Cornell University, September 2003.

[16] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. *IEEE International Conference on Computer Vision*, July 2001.

[17] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. Technical report, Cornell University, 2001.

[18] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts ? *European Conference on Computer Vision*, May 2002.

[19] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47, 2002.

[20] Heung-Yeung Shum, Jian Sun, and Nan-Ning Zheng. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25, July 2003.

[21] Petter Strandmark and Fredrik Kahl. Parallel and distributed graph cuts by dual decomposition. In *Conference on Computer Vision and Pattern Recognition*, 2010.

[22] Zeng-Fu Wang and Zhi-Gang Zheng. A region based stereo matching algorithm using cooperative optimization. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[23] Qingxiong Yang, Liang Wang, Ruigang Yang, Henrik Stewénius, and David Nistér. Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31, March 2009.

[24] C. Lawrence Zitnick and Takeo Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, July 2000.

Index

assignement, 20
  assignement actif, 20
  assignement inactif, 20

baseline, 8
benchmark, 14

caméra, 9
configuration, 11
  configuration correspondant à une coupe, 26
  configuration unique, 20

disparité, 8
  décalage d'un assignement, 21
  disparité d'un pixel, 21

expansion move, 26

fonctionnelle d'énergie
  condition de régularité, 24
  représentable par un graphe, 23
  terme de données, 21
  terme de fidélité, 21
  terme de occlusion, 21
  terme de régularité, 21

géométrie épipolaire, 10
graphe, 15
  algorithme de Ford–Fulkerson, 17
  arête, voir arc
  arbre, 48
    arbre enraciné, 48
    enfant, 48
    forêt, 48
    parent, 48
    racine, 48
  arc, 15
    arc aboutissant, 15
    arc partant, 15
  capacité, 15
    capacité résiduelle, 16
    constrainte de capacité, 15
  chaîne, 16
    chaîne augmentante, 17
    chaîne saturée, 17
  coupe, 16
    capacité, 16
    coupe minimale, 16
  flot, 15
    flot maximal, 15
    flot réalisable, 15
    propriété de conservation, 15
    valeur, 15
  graphe orienté, 15
  nœud, 15
    actif, 17
    inactif, 17
    libre, 17
    orphelin, 18
  puits, 15
  réseau, 15
  sommet, voir nœud
  source, 15
  théorème de Max-Flow/Min-Cut, 17

image
  image d'un point, 9
  image de référence, 11
  plan image, 9

méthode
  méthodes globales, 11
  méthodes locales, 12
mesure de dissimilarité de Birchfield–Tomasi, 34
mise en correspondance, 11

occlusion
  désocclusion, 11
  phénomène d'occlusion, 10
  pixels occlus, 10

paramètre, 22
  choix des paramètres, 32

rectification, 10

système de voisinage, 22

vérité-terrain, 7

Annexe A : Arbres

On introduit dans cette annexe quelques notions élémentaires sur les arbres.

Définition a (Chemin élémentaire) *Dans un graphe orienté, on appelle* chemin élémentaire *un chemin (une suite d'arcs consécutifs) qui visite chaque nœud au plus une fois.*

On peut alors définir un arbre :

Définition b (Arbre) *On appelle* arbre *un graphe orienté dans lequel chaque paire de sommet est relié par un chemin élémentaire unique.*
Une forêt *est un ensemble d'arbres.*

Définition c (Parent, enfant) *Soit (x, y) un arc dans un arbre. Le nœud x est appelé* parent *de y et réciproquement, y est un* enfant *de x.*

Un *arbre enraciné* est alors un arbre tel qu'il n'existe qu'un seul nœud qui ne possède aucun parent ; ce nœud est appelé *racine* de l'arbre.

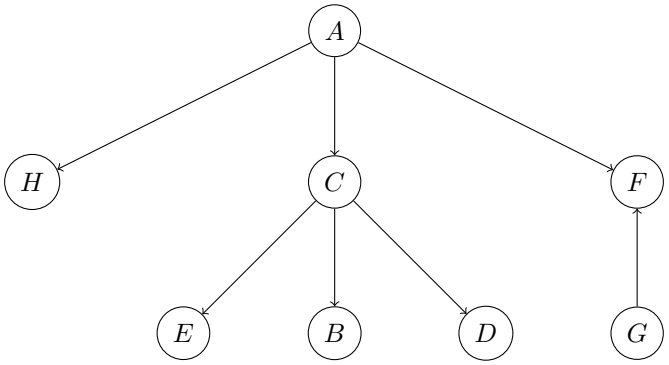


FIGURE 35 – Exemple d'arbre (enraciné). Les enfants du sommet C sont E, B et D. La racine de cet arbre est A.

Annexe B : Preuve de la représentation de la fonctionnelle d'énergie par le graphe proposé

On prouve dans cette annexe les trois théorèmes énoncés dans la partie IV.5.

Lemme d *$\mathcal{C}$ est une coupe sur $\mathcal{G}$ si, et seulement si, la configuration $f^{\mathcal{C}}$ correspondante est une single α-expansion move de f°.*

Preuve : ($\Rightarrow$) Si $\mathcal{C}$ est une coupe sur $\mathcal{G}$, alors soit $\mathcal{A}^{\mathcal{C}} = A(f^{\mathcal{C}})$ l'ensemble des assignements actifs de $f^{\mathcal{C}}$; soit $a \in \mathcal{A}^{\mathcal{C}}$. Si $a \in \mathcal{V}^s$, alors a a la même disparité dans la configuration f° que dans la configuration $f^{\mathcal{C}}$; sinon, $d(a) = \alpha$. D'où $f^{\mathcal{C}}$ est bien une single α-expansion move de f°.

($\Leftarrow$) On suppose que $\mathcal{V}^s, \mathcal{V}^t \subset \mathcal{G}$. On a alors $\mathcal{V}^s \cap \mathcal{V}^t = \varnothing$ et $\mathcal{V}^s \cup \mathcal{V}^t = \mathcal{V}$ car sinon, $f^{\mathcal{C}}$ n'est pas bien définie. D'où $(\mathcal{V}^s, \mathcal{V}^t)$ définit bien une coupe de $\mathcal{G}$. ■

Lemme e *Le coût de la coupe $\mathcal{C}$ est fini si, et seulement si, la configuration $f^{\mathcal{C}}$ correspondante est unique.*

Preuve : ($\Rightarrow$) Si la configuration n'est pas unique, il existe $p \in \mathcal{P}$ et a_1, a_2 deux assignements actifs impliquant p ; dans ce cas, p apparaît deux fois dans le graphe, donc il y a un arc de poids infini qui relie ces deux sommets, et comme $\mathcal{C}$ est de coût fini, $a_1, a_2 \in \mathcal{V}^s$ et $a_1, a_2 \in \mathcal{V}^t$; contradiction. D'où la configuration est unique.

($\Leftarrow$) On suppose la configuration unique et le coût de $\mathcal{C}$ infini ; il existe donc un arc de poids infini reliant un sommet $a_1 \in \mathcal{V}^s$ et $a_2 \in \mathcal{V}^t$; donc, a_1 et a_2 ont un pixel en commun, ce qui implique que $a_1 \in \mathcal{A}^\circ$ et $a_2 \in \mathcal{A}^\alpha$ ou vice-versa. Donc $f^{\mathcal{C}}_{a_1} = f^{\mathcal{C}}_{a_2} = 1$, ce que contredit le fait que la configuration est unique. ■

On appellera c-arêtes les arcs de poids $2\,K$ et t-arêtes les arcs de poids $D_{\text{occlusion}}(a)$.

Lemme f *Soit $f^{\mathcal{C}}$ une configuration unique, correspondant à la coupe $\mathcal{C}$. Alors le coût des t-arêtes et des c-arêtes dans $\mathcal{C}$ est égal à $E_{\text{occlusion}}(f^{\mathcal{C}})$ plus une constante.*

On ajoute, dans le graphe $\mathcal{G}$, une étiquette aux sommets, valant 1 si le sommet est actif dans la configuration $f^{\mathcal{C}}$ et 0 sinon.

Preuve : $f^{\mathcal{C}}$ est une configuration unique, donc le coût de la coupe $\mathcal{C}$ est fini. En particulier, cela implique que $\mathcal{C}$ ne contient aucune c-arête (car sinon, elle contiendrait aussi une arête de poids infini, puisque si elle contient (a_1, a_2), elle contient également (a_2, a_1)). Il suffit donc de considérer le coût des t-arêtes.
On remarque que les t-arêtes de $\mathcal{C}$ sont précisément les arêtes qui relient s aux 0 de $\mathcal{A}^\circ$ et les 0 de $\mathcal{A}^\alpha$ à t ; en effet, les t-arêtes sont les arcs (s, a) pour $a \in \mathcal{A}^\circ$ et (a, t) pour $a \in \mathcal{A}^\alpha$, tandis que les t-arêtes de $\mathcal{C}$ sont celles qui relient s à un sommet dans $\mathcal{V}^t \cap \mathcal{A}^\circ$ (qui est donc étiqueté 0) et un sommet dans $\mathcal{V}^s \cap \mathcal{A}^\alpha$ (donc étiqueté 0 également) à t.
Calculons $E_{\text{occlusion}}(f^{\mathcal{C}})$:

$$E_{\text{occlusion}}(f^{\mathcal{C}}) = \sum_{p \in \mathcal{P}} 2\,K \cdot T(|A_p(f^{\mathcal{C}})| = 0)$$

Puisque $\tilde{A} = \mathcal{A}^\circ \cup \mathcal{A}^\alpha = A(\tilde{f})$, tous les sommets sont actifs pour $\tilde{f}$; par ailleurs, on sait que p ne peut apparaître au plus que deux fois dans le graphe, donc $1 \leq |A_p(\tilde{f})| \leq 2$; on peut donc séparer la somme en trois parties :

$$\begin{aligned} E_{\text{occlusion}}(f^{\mathcal{C}}) = &\sum_{\substack{p \in \mathcal{P} \\ |A_p(\tilde{f})|=0}} 2\,K \cdot T(|A_p(f^{\mathcal{C}})| = 0) + \sum_{\substack{p \in \mathcal{P} \\ |A_p(\tilde{f})|=1}} 2\,K \cdot T(|A_p(f^{\mathcal{C}})| = 0) \\ &+ \sum_{\substack{p \in \mathcal{P} \\ |A_p(\tilde{f})|=2}} 2\,K \cdot T(|A_p(f^{\mathcal{C}})| = 0) \end{aligned}$$

le premier terme correspond aux pixels qui sont occlus pour $\tilde{f}$, qui est donc constant par rapport à la coupe ; on le notera K.
Si $|A_p(\tilde{f})| = 2$, $|A_p(f^{\mathcal{C}})| = 1$ car p apparaît deux fois dans le graphe (une fois dans $\mathcal{A}^\circ$ et l'autre dans $\mathcal{A}^\alpha$), donc une fois dans un sommet étiqueté 1 car ces deux sommets sont alors reliés par un arc de poids infini. D'où le terme d'occlusion devient :

$$E_{\text{occlusion}}(f^{\mathcal{C}}) = K + \sum_{\substack{p \in \mathcal{P} \\ |A_p(\tilde{f})|=1}} 2\,K \cdot T(|A_p(f^{\mathcal{C}})| = 0)$$

i.e. on somme sur les pixels occlus pour $f^{\mathcal{C}}$ et qui n'apparaissent qu'une seule fois dans le graphe, c'est-à-dire exactement

$$E_{\text{occlusion}}(f^{\mathcal{C}}) = K + \sum_{a \in \tilde{A}} D_{\text{occlusion}}(a) \cdot T(a \notin A(f^{\mathcal{C}}))$$

d'où le résultat. ■

Théorème g *Soit $\mathcal{C}$ la coupe minimale de $\mathcal{G}$. Alors $f^{\mathcal{C}}$ est le α-expansion move de f° qui minimise l'énergie E.*

Preuve : Soit $\mathcal{C}$ une coupe de $\mathcal{G}$; calculons le coût de cette coupe :

$$\begin{aligned}
\mathrm{cap}(\mathcal{C}) &= E_{\text{occlusion}}(f^{\mathcal{C}}) - K + \sum_{\substack{a\in\mathcal{A}^\circ \\ f_a^{\mathcal{C}}=1}} (D(a) + D_{\text{régularité}}(a)) + \sum_{\substack{a\in\mathcal{A}^\alpha \\ f_a^{\mathcal{C}}=1}} D(a) \\
&\qquad + \sum_{\substack{\{a_1,a_2\}\in\tilde{A} \\ a_1,a_2\in\mathcal{N}}} V_{a_1,a_2}\cdot T(f_{a_1}^{\mathcal{C}} \neq f_{a_2}^{\mathcal{C}}) \\
&= E_{\text{occlusion}}(f^{\mathcal{C}}) - K + \sum_{a\in\tilde{A}} D(a)\cdot T(f_a^{\mathcal{C}}=1) + \sum_{a\in\mathcal{A}^\circ} D_{\text{régularité}}(a)\cdot T(f_a^{\mathcal{C}}=1) \\
&\qquad + \sum_{\substack{\{a_1,a_2\}\in\tilde{A} \\ a_1,a_2\in\mathcal{N}}} V_{a_1,a_2}\cdot T(f_{a_1}^{\mathcal{C}} \neq f_{a_2}^{\mathcal{C}}) \\
&= E_{\text{occlusion}}(f^{\mathcal{C}}) - K + \sum_{a\in\tilde{A}} D(a)\cdot T(f_a^{\mathcal{C}}=1) + \sum_{\substack{a1\in\mathcal{A}^\circ, a_2\notin\tilde{A} \\ a_1,a_2\in\mathcal{N}}} V_{a_1,a_2}\cdot T(f_{a_1}^{\mathcal{C}}=1) \\
&\qquad + \sum_{\substack{\{a_1,a_2\}\in\tilde{A} \\ a_1,a_2\in\mathcal{N}}} V_{a_1,a_2}\cdot T(f_{a_1}^{\mathcal{C}} \neq f_{a_2}^{\mathcal{C}}) \\
&= E_{\text{occlusion}}(f^{\mathcal{C}}) - K + \sum_{a\in\tilde{A}} D(a)\cdot T(f_a^{\mathcal{C}}=1) + \sum_{\substack{a1\in\mathcal{A}^\circ, a_2\notin\tilde{A} \\ a_1,a_2\in\mathcal{N}}} V_{a_1,a_2}\cdot T(f_{a_1}^{\mathcal{C}} \neq f_{a_2}^{\mathcal{C}}) \\
&\qquad + \sum_{\substack{\{a_1,a_2\}\in\tilde{A} \\ a_1,a_2\in\mathcal{N}}} V_{a_1,a_2}\cdot T(f_{a_1}^{\mathcal{C}} \neq f_{a_2}^{\mathcal{C}}) \\
\mathrm{cap}(\mathcal{C}) &= E_{\text{occlusion}}(f^{\mathcal{C}}) - K + \sum_{a\in\tilde{A}} D(a)\cdot T(f_a^{\mathcal{C}}=1) \\
&\qquad + \sum_{a_1,a_2\in\mathcal{N}} V_{a_1,a_2}\cdot T(f_{a_1}^{\mathcal{C}} \neq f_{a_2}^{\mathcal{C}})
\end{aligned}$$

car si $a_1 \in \mathcal{A}^\alpha$ et $\{a_1, a_2\} \in \mathcal{N}$, on a forcément $a_2 \in \mathcal{A}^\alpha$. Par ailleurs, puisque $A(f^{\mathcal{C}}) \subset \tilde{A}$, on a :

$$\sum_{a\in\tilde{A}} D(a)\cdot T(f_a^{\mathcal{C}}=1) = \sum_{a\in A(f^{\mathcal{C}})} D(a)\cdot T(f_a^{\mathcal{C}}=1).$$

D'où finalement, on a donc $E(f^{\mathcal{C}}) = \mathrm{cap}(\mathcal{C}) + K$. D'où le minimum de cette énergie sur les coupes de $\mathcal{G}$ est atteint par la coupe minimale. ■