Optimisation à base de flot de graphe pour l'acquisition d'informations 3D à partir de séquences d'images

Sylvain Paris — François Sillion

ARTIS - GRAVIR / IMAG - INRIA [1]
655, avenue de l'Europe, Montbonnot
38334 Saint Ismier cedex
{sylvain.paris|francois.sillion}@imag.fr

RÉSUMÉ. *On se propose de montrer comment on peut appliquer la technique de « flot de graphe » pour résoudre certains problèmes complexes d'optimisation. Cette technique est utilisée ici pour l'acquisition d'informations 3D à partir de plusieurs points de vue. On montre notamment comment – grâce à un graphe particulier – il est possible de minimiser une famille d'énergies qui permettent d'atteindre des résultats plus précis que les cartes de disparité obtenues jusqu'à présent sur ce type de problème. Cette nouvelle approche offre aussi la possibilité de prendre en compte les discontinuités des objets avec lesquels on souhaite travailler. Les résultats ainsi obtenus sont exposés.*

ABSTRACT. *We show how the graph flow technique can be used to solve some complex optimization problems. This technique is used here to acquire 3D information using several points of view. We show how – with an adapted graph – it is possible to minimize a category of energies which can reach more precise results than the usual disparity maps. This novel approach also takes object discontinuities into account. The results obtained with the method are exposed.*

MOTS-CLÉS : *flot de graphe, coupure de graphe, optimisation, reconstruction 3D.*

KEYWORDS: *graph flow, graph cut, optmization, 3D reconstruction.*

1. ARTIS est une équipe du laboratoire GRAVIR/IMAG, unité mixte de recherche CNRS, INPG, INRIA, UJF.

1. Introduction

Nous nous sommes intéressés au problème de l'acquisition d'informations tridimensionnelles à partir d'une courte séquence d'images. Notre objectif est de « remplir » des environnements virtuels auxquels manquent la plupart du temps les petits détails essentiels au réalisme comme une boîte aux lettres, un banc public ou des passants. Ces détails sont à la fois nombreux et fastidieux à modéliser à la main – il ne semble par exemple pas raisonnable de modéliser l'intégralité du mobilier urbain d'une ville. Ces éléments sont pourtant indispensables à la qualité d'ensemble d'un environnement virtuel. C'est pour cela que nous nous sommes posé la question de la reconstruction 3D à partir d'une séquence d'images.

Un des points cruciaux de ce problème de reconstruction 3D est la localisation de la surface des objets. Celle-ci doit bien évidemment être cohérente par rapport aux images données en entrée. Mais ce critère ne suffit pas car le problème de la reconstruction est intrinsèquement mal posé et qui plus est, on souhaite exploiter des images réelles qui peuvent éventuellement présenter des imperfections. On impose donc en plus de cette cohérence un critère de continuité : la plupart des objets ont une surface régulière – au moins localement. Ce type de compromis se représente classiquement sous forme d'une fonction d'énergie qui contient un terme en rapport avec la régularité et un terme en rapport avec la cohérence. Pour le terme de régularité nous avons choisi une pénalité sur les dérivées de la surface et nous obtenons le terme de cohérence des étapes précédentes de notre algorithme qui est décrit dans [PAR 02]. Or il se trouve que ce terme a a priori très peu de propriétés mathématiques qui peuvent servir de base aux techniques classiques d'optimisation. Les techniques à base de flot de graphe sont alors apparues comme une solution à ce problème. Nous allons donc présenter une technique d'optimisation qui permet de trouver la fonction minimisante d'une énergie pour une famille d'énergies qui englobe notre problème.

Dans la section 2 nous allons observer les principaux travaux existants sur le problème de la reconstruction 3D pour examiner quelle technique d'optimisation est employée en nous attardant sur les méthodes à base de flot de graphe. Ensuite dans la section 3 nous introduirons la notion de flot de graphe. Nous continuerons dans la section 4 en présentant comment cette notion peut être utile pour résoudre des problèmes d'optimisation. Dans la section 5 nous étudierons l'application de cette technique à notre problème d'acquisition d'informations 3D. Nous finirons par la section 6 où l'on présente quelques résultats et par la section 7 qui conclut cet article.

2. Travaux existants

Toute une famille de méthodes de reconstruction est basée sur la discrétisation de l'espace en voxels ; la plus connue est le *Space Carving* introduit par Kutulakos et Seitz [KUT 99] dont de nombreuses variantes ont été dérivées. Pour plus de détails, on pourra lire le tour d'horizon proposé par Slabaugh *et al.* [SLA 01] qui regroupe la plupart d'entre elles. Pour toutes ces méthodes, l'évaluation de la présence ou non d'un voxel se fait uniquement selon la cohérence par rapport aux images, aucune contrainte de régularité n'est prise en compte il n'y a donc aucune optimisation d'effectuée. Ces techniques sont donc très sensibles à la qualité des images initiales et à l'ambiguïté due au faible angle de vue dont nous disposons dans nos courtes séquences d'images.

On peut ensuite citer les techniques travaillant avec les lignes épipolaires [FAU 01, HAR 00] pour créer des cartes de disparité [OKU 93, OHT 85, INT 94, ULV 98]. Ces méthodes introduisent naturellement des contraintes de régularité le long des lignes épipolaires et obtiennent ainsi des résultats même pour un faible angle de vue. Néanmoins elles présentent deux restrictions principales : il est difficile d'introduire une régularité entre deux lignes épipolaires et les cartes de disparité obtenues ont une faible précision en profondeur, les objets reconstruits sont généralement plats. Koch *et al.* [KOC 98] améliorent la précision en utilisant un filtre de Kalman mais n'introduisent toujours pas de régularité entre lignes épipolaires.

Faugeras et Keriven [FAU 98] proposent une méthode à base de courbes de niveaux qui fait évoluer une surface dans l'espace de manière à s'approcher de plus en plus de la surface de l'objet à reconstruire. Grâce aux courbes de niveaux, cette technique optimise un compromis entre la cohérence par rapport aux images et la régularité de la surface finale. Néanmoins, elle nécessite des images de très bonne qualité et des points de vue très distants pour obtenir des résultats satisfaisants.

Viennent ensuite les méthodes à base de flots de graphes. Roy et Cox [ROY 98] ont introduit leur technique comme une généralisation des méthodes travaillant sur les lignes épipolaires pour construire des cartes de disparités. Leur méthode montre qu'un flot de graphe permet d'étendre le processus d'optimisation à toute l'image. Ils constatent aussi que leur processus a des propriétés de régularisation similaires aux techniques à base de lignes épipolaires. Cette technique a été par la suite formalisée par Veksler [VEK 99] puis Kolmogorov et Zabih [KOL 01, KOL 02a, KOL 02b] sous la forme d'un problème de labellisation visant à construire une carte de disparité. Chaque label est une valeur de disparité possible. Un label est associé à chaque pixel de manière à minimiser une énergie qui tient compte du voisinage du pixel pour avoir un résultat final régulier. Ces méthodes ont deux limitations, la première est qu'elles construisent des cartes de disparité qui ont toujours le défaut d'aplatir les objets ; la seconde est que, comme la fonction de pénalité entre deux voisins de labels différents n'est pas nécessairement convexe, la minimisation de l'énergie est un problème NP-complet dont on obtient finalement qu'une approximation. Par contre, Ishikawa [ISH 00] propose l'étude du cas où cette fonction est convexe et décrit un graphe qui permet d'obtenir le résultat exact. Toutefois la méthode de stéréovision qu'il expose ne peut pas se généraliser aisément à une séquence d'images et n'utilise qu'une pénalité linéaire. Une méthode pour utiliser trois caméras a été proposée par Buehler *et al.* [BUE 02] mais celle-ci ne s'étend pas non plus à une séquence d'images.

Contribution

La technique que nous allons décrire se propose de formuler le problème d'optimisation sous une forme nouvelle qui exhibe le rôle de la configuration géométrique et d'y adapter une méthode de flot de graphe qui tire partie des fonctions de pénalité convexes. Ainsi il sera possible de travailler à partir de séquences d'images et d'obtenir des modèles plus précis que les cartes de disparités produites jusqu'à présent.

3. Flot de graphe

Le problème du flot de graphe est un problème classique d'algorithmique. Initialement, il s'agit de la formulation d'un problème simple d'écoulement d'eau dans un réseau de tuyaux. On présente dans un premier temps ce problème pour donner l'intuition de ce que représente la formulation théorique qui suit.

3.1. *Écoulement d'eau dans un réseau*

Le problème que l'on se pose est le suivant. Étant donnés une source d'eau, un puits et un réseau de tuyaux reliant la source au puits, on cherche le flot maximum que l'on peut faire passer à travers le réseau. On suppose que la source et le puits peuvent fournir et recevoir autant d'eau que l'on souhaite, le flot maximum est alors uniquement contraint par le réseau. Intuitivement, on peut voir le réseau comme un barrage entre la source et le puits qui ne laisse passer qu'un certain débit.

Si le réseau restreint le flot, c'est qu'il contient un goulot d'étranglement. Considérons un ensemble de tuyaux qui sépare la source du puits. Pour aller de la source au puits, l'eau doit nécessairement emprunter l'un de ces tuyaux. Donc dans le meilleur des cas, si tous ces tuyaux sont pleins, le flot sera égal à la somme de leurs capacités. Le goulot d'étranglement correspond alors à un ensemble de tuyaux qui sépare la source du puits et dont la somme des capacités est minimale.

Remarquons, pour finir avec cet exemple, que si le flot est maximum à travers le réseau, on est dans le cas où tous les tuyaux du goulot d'étranglement sont pleins. La valeur du flot maximum est par conséquent égale à la capacité minimale d'un ensemble séparateur. Trouver l'une ou l'autre de ces valeurs sont donc deux problèmes liés.

3.2. *Formulation théorique*

On considère un graphe orienté connexe $\mathcal{G}$ composé d'un ensemble de sommets $\mathcal{S}$ et d'un ensemble d'arcs orientés $\mathcal{A} \in \mathcal{S}^2$. On distingue deux sommets particuliers : la *source* s et le *puits* p. Pour un sommet x, on définit l'ensemble des arcs *entrants* $\mathcal{A}_e(x)$ et celui des arcs *sortants* $\mathcal{A}_s(x)$:

$$\mathcal{A}_e(x) = \{a \in \mathcal{A} \ / \ \exists y \in \mathcal{S}, a = (y, x)\}$$
$$\mathcal{A}_s(x) = \{a \in \mathcal{A} \ / \ \exists y \in \mathcal{S}, a = (x, y)\}$$

On définit une fonction *capacité* qui associe à un arc a un réel positif $Cap(a)$ et une fonction *flot* qui lui associe un autre réel positif $Flot(a)$. On dit que le flot est *valide* si :

$$\forall a \in \mathcal{A} \qquad Flot(a) \leq Cap(a) \qquad [1]$$

$$\forall x \in \mathcal{S}\backslash\{s,p\} \qquad \sum_{a_e \in \mathcal{A}_e(x)} Flot(a_e) = \sum_{a_s \in \mathcal{A}_s(x)} Flot(a_s) \qquad [2]$$

s et p étant exclus de la contrainte 2, pour simplifier la suite des définitions, on suppose $\mathcal{A}_e(s) = \mathcal{A}_s(p) = \emptyset$.

Dans l'exemple précédent, $\mathcal{G}$ est le réseau, $\mathcal{A}$ l'ensemble des tuyaux et $\mathcal{S}$ l'ensemble des jonctions entre tuyaux. La fonction de capacité indique le débit maximum dans un tuyau, celle de flot donne le débit effectif dans un tuyau. Le flot est valide si toute l'eau qui arrive à une jonction en repart (équation [2]) et si le flot dans un tuyau n'est pas supérieur à son débit maximum (équation [1]).

À partir de maintenant, on ne considère plus que des flots valides.

À une fonction de flot donnée, on associe une valeur que l'on appellera *flot de graphe* (ou simplement *flot* si cela ne prête pas à ambiguïté) :

$$Flot(\mathcal{G}) = \sum_{a \in \mathcal{A}_s(s)} Flot(a) \qquad [3]$$

On définit une *coupure* $\mathcal{C}_\mathcal{G}$ de $\mathcal{G}$ comme la partition de $\mathcal{S}$ en deux ensembles connexes $\mathcal{S}_s$ et $\mathcal{S}_p$ tels que $s \in \mathcal{S}_s$ et $p \in \mathcal{S}_p$. On associe à cette coupure une valeur $Coup(\mathcal{C}_\mathcal{G})$ (que l'on appellera aussi *coupure* quand cela ne crée pas d'ambiguïté) :

$$Coup(\mathcal{C}_\mathcal{G}) = \sum_{(x,y) \in (\mathcal{S}_s \times \mathcal{S}_p)} Cap(x,y) \qquad [4]$$

et on dira qu'un arc (x,y) est *coupé* si $(x,y) \in (\mathcal{S}_s \times \mathcal{S}_p)$.

Par la suite, on s'intéressera à la coupure de valeur minimale. Pour cela, on a le théorème suivant qui constituera la base de la méthode proposée.

Théorème « Flot maximum - coupure minimale » : Étant donnés un graphe $\mathcal{G}$, une source s, un puits p et une fonction de capacité Cap, on a la relation suivante entre le flot maximal atteint sur l'ensemble des fonctions de flot valides et la coupure de valeur minimale :

$$\max Flot(\mathcal{G}) = \min Coup(\mathcal{C}_\mathcal{G}) \qquad [5]$$

Par rapport à l'exemple initial, ce théorème est simplement la formalisation du fait qu'il ne peut pas passer plus d'eau à travers le réseau que le goulot d'étranglement ne le permet.

On trouvera la démonstration du théorème dans [FOR 62, COR 90]. L'intérêt majeur de ce résultat est de montrer qu'en utilisant des algorithmes qui calculent le flot maximum [GOL 97, CHE 97] on a accès à la coupure minimale.

4. Minimisation d'énergie

L'idée à la base de la méthode que l'on propose est de créer un graphe de manière à ce qu'une coupure représente une fonction et que la valeur de cette coupure soit l'énergie associée à la fonction. En trouvant la coupure minimale, on aura donc calculé une fonction minimisante. On commence par un exemple simple pour illustrer le concept. On étudie ensuite les cas à deux dimensions et à trois dimensions.

4.1. *Exemple simple*

Prenons trois points x_1, x_2 et x_3, une fonction f qui peut prendre deux valeurs y_1 et y_2. Pour définir une énergie sur f, on introduit une fonction $c(x, y) > 0$ qui représente le coût du choix $f(x) = y$ et une fonction de pénalité $p(f(x_i), f(x_{i+1}))$ pour $i \in \{1, 2\}$ qui représente notre souhait d'avoir des valeurs de f similaires pour des points proches. $p(f(x_i), f(x_{i+1}))$ est nulle si $f(x_i) = f(x_{i+1})$ et vaut $p_0 > 0$ sinon.

On cherche f qui minimise :

$$\mathcal{E}(f) = \sum_{i=1}^{3} c(x_i, f(x_i)) + \sum_{i=1}^{2} p(f(x_i), f(x_{i+1})) \qquad [6]$$

On construit un graphe tel qu'indiqué sur la figure 1. La base du graphe est une grille dont les arcs sont bidirectionnels. À chaque position (x, y) on associe un arc vertical avec la capacité $c(x, y)$, les arc horizontaux correspondent à la fonction de pénalité. Les arcs qui relient la source à la grille et la grille au puits ont une capacité infinie.

Si on étudie maintenant une coupure pour ce graphe (figure 1 à droite), on peut faire les remarques suivantes :

– si elle coupe un arc qui est lié à la source ou au puits, elle aura une valeur infinie donc ne sera pas minimale ;

– si on exclut ces coupures infinies, comme une coupure crée une partition des sommets, elle doit forcément passer par au moins un arc dont l'abscisse est x_i pour $i \in \{1, 2, 3\}$;

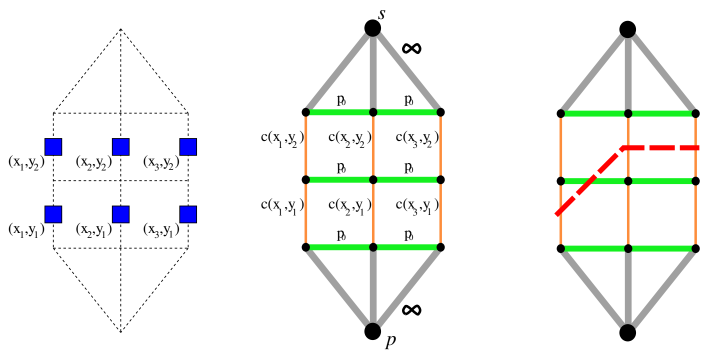

Figure 1. *Un exemple simple. À gauche, les positions possibles pour f et la structure du graphe en pointillés (les positions sont associées à des arcs verticaux). Au milieu, le graphe avec les capacités des arcs. À droite, un exemple de coupure*

– une coupure minimale ne coupe qu'un seul arc vertical d'abscisse x_i car une coupure coupant deux tels arcs aura une valeur plus élevée que si elle ne coupait que l'un des deux (le détail de ce point est laissé au lecteur).

Une coupure qui satisfait ces trois remarques est dite *potentiellement minimale* : elle ne coupe aucun arc infini et ne coupe qu'un et un seul arc d'abscisse x_i.

Par conséquent, on peut construire une fonction f à partir d'une coupure potentiellement minimale : à partir de chaque arc vertical (x_i, y_j) on déduit un point $f(x_i) = y_j$. Inversement, à partir d'une fonction f, on construit une coupure potentiellement minimale en coupant uniquement les arcs verticaux (x_i, y_j) tels que $f(x_i) = y_j$ (les arcs horizontaux coupés s'en déduisent directement). Observons pour finir la valeur d'une telle coupure : les arcs verticaux coupés forment exactement la somme $\sum_{i=1}^{3} c(x_i, f(x_i))$ et les arcs horizontaux coupés la somme $\sum_{i=1}^{2} p(f(x_i), f(x_{i+1}))$ de l'énergie (6).

En conclusion, en calculant la coupure minimale du graphe en figure 1, on trouve la fonction $f_0 = \operatorname{argmin}_f \mathcal{E}(f)$.

Avant de passer à un cas plus complexe, observons quelques points importants.

– Nous avons un calcul direct du résultat, nous n'avons utilisé aucune technique type descente de gradient susceptible de se bloquer dans un minimum local. Nous sommes assurés d'avoir le minimum global de l'énergie.

– On peut exprimer la pénalité p_0 sous la forme $\alpha\,|\Delta f(x_i)|$, ce qui offre une interprétation géométrique de la pénalité : on pénalise proportionnellement à la variation de f.

– La fonction de pénalité p et le coefficient α peuvent aussi être fonction de x_i. Cela permet d'introduire des contraintes plus souples, on peut affecter par exemple une pénalité plus importante au cas $f(x_1) \neq f(x_2)$ qu'au cas $f(x_2) \neq f(x_3)$, ce qui se traduit par $\alpha(x_1) > \alpha(x_2)$.

4.2. *Étude en deux dimensions*

Après cet exemple simple, nous pouvons étendre la méthode à une fonction plus générale en deux dimensions. Il suffit de prendre en compte plus de valeurs pour x et y. Pour cela, on va poser un problème basé sur une fonctionnelle continue et la discrétiser selon les deux axes.

Considérons une fonction f réelle d'un intervalle $[a,b]$ dans un intervalle $[c,d]$, une fonction de coût $c(x,y)$ et un coefficient de pénalité $\alpha(x)$. On définit de manière similaire une énergie $\mathcal{E}$ qui est d'autant plus faible que la fonction f prend des valeurs de faible coût tout en ayant des valeurs voisines proches les unes des autres.

$$\mathcal{E}(f) = \int_a^b \left(c(x, f(x)) + \alpha(x) \left| \frac{df}{dx}(x) \right| \right) dx \qquad [7]$$

Nous allons montrer comment trouver une fonction f_0 qui minimise cette énergie à une discrétisation près. En effet, comme nous utilisons des graphes, nous ne pouvons traiter que des cas discrets. Néanmoins, il est toujours possible de discrétiser plus finement pour obtenir des résultats plus précis. Nous découpons par conséquent l'intervalle $[a,b]$ en $n_x - 1$ sous-intervalles de même longueur Δx pour obtenir n_x points $x_1, x_2, \ldots x_{n_x}$. Nous faisons de même pour $[c,d]$ pour obtenir n_y points $y_1, y_2, \ldots y_{n_y}$ espacés de Δy. L'énergie discrète correspondant à (7) est alors :

$$\mathcal{E}^d(f) = \sum_{i=1}^{n_x} c(x_i, f(x_i))\Delta x + \sum_{i=1}^{n_x - 1} \alpha(x_i) \left| \Delta f(x_i) \right| \qquad [8]$$

On obtient une énergie très similaire à celle obtenue en (6). On procède par conséquent de la même manière : on utilise un graphe basé sur une grille (figure 2 à gauche) où les arcs verticaux représentent la fonction de coût $c(x,y)\Delta x$ et les arcs horizontaux la pénalité $\alpha(x)\Delta y$ pour les valeurs distinctes. On introduit de manière totalement similaire les *coupures potentiellement minimales* (en remarquant que l'unicité de la valeur déterminée est assurée par le fait que tous les arcs horizontaux de même abscisse ont la même valeur) et on leur associe à chacune une fonction f. La conclusion étant que la fonction f_0 associée à la coupure minimale réalise : $f_0 = \text{argmin}_f \mathcal{E}^d$.

Observons à nouveau quelques détails sur la méthode.

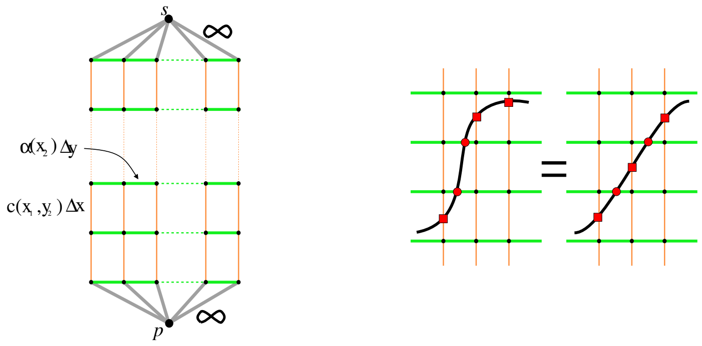

Figure 2. *À gauche : graphe correspondant à l'énergie (8). À droite : si le coût et la pénalité sont uniformes, il y a ambiguïté. Les deux coupures ont exactement la même valeur : trois arcs de coût (les carrés) et 2 arcs de pénalité (les ronds)*

– Ce problème se résout très bien grâce à la programmation dynamique [BEL 57], il ne s'agit encore que d'un cas d'étude.

– La formulation (7) exhibe la géométrie du problème : l'élément d'intégration dx explicite la mesure employée. Si on change la configuration géométrique – par un changement d'échelle par exemple – il est aisé de recalculer les différentes grandeurs de (8) en fonction de cette nouvelle configuration.

– La seule contrainte pour la fonction de coût est de pouvoir la discrétiser donc qu'elle soit continue par morceaux.

– En regardant l'énergie (7), la fonction f_0 doit être continûment dérivable sauf sur les points où $\alpha(x) = 0$.

– Comme précédemment, on trouve un minimum global de manière certaine.

– On a montré la méthode sur un domaine 2D rectangulaire mais il est tout à fait possible de travailler sur un domaine plus général. Il suffit alors de discrétiser ce domaine selon une grille et de relier les nœuds de la limite supérieure à la source et ceux de la limite inférieure au puits.

– Comme la pénalité est simplement proportionnelle à la pente de la coupure, dans les zones où les fonctions de coût et de pénalité sont constantes, on peut trouver plusieurs coupures de même valeur car un palier ou une pente « douce » ont la même énergie (figure 2 à droite). A priori, vu que l'on impose une pénalité aux valeurs distinctes, on souhaite obtenir la pente douce alors qu'en pratique les algorithmes donnent toujours le palier.

Une pénalité non linéaire

La dernière remarque soulève un problème important : on obtient des paliers malgré l'introduction dans l'énergie d'une composante visant à limiter les variations de la fonction solution. La linéarité de la pénalité est à l'origine de ce phénomène car elle ne différencie pas une importante variation de plusieurs petites. Nous proposons par conséquent une structure de graphe introduisant une pénalité approchant la pénalité initiale tout en étant strictement convexe : il sera alors plus pénalisant pour la fonction de faire un palier que de faire une variation régulière. Ishikawa [ISH 00] propose une méthode générale pour obtenir une fonction de pénalité convexe à une constante près sur la pénalité, ce qui peut être gênant dans le cas d'un échantillonnage irrégulier. Nous proposons une autre méthode qui peut représenter moins de fonctions de pénalité (toutes les fonctions convexes ne sont pas accessibles) mais n'introduit pas de constante et suffit à notre objectif.

Le graphe est construit à partir de l'élément de base de la figure 3 : l'arc vertical de coût est remplacé par quatre arcs *mineurs* de capacité moitié et on introduit un nouveau coefficient de pénalité β dit *secondaire*. Il correspond à la capacité des arcs de pénalités reliant les milieux de deux arcs verticaux adjacents. On appellera α le coefficient *principal* pour éviter les ambiguïtés.

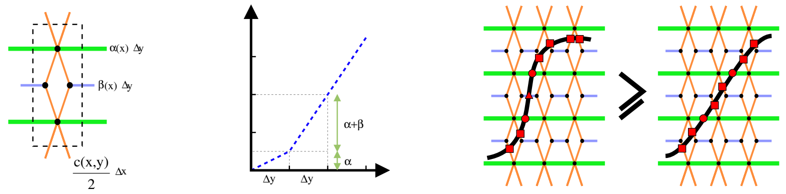

Figure 3. *À gauche : l'élément de base utilisé. Au milieu : la fonction de pénalité* p_{conv} *correspondante. À droite : une variation continue est moins pénalisée qu'un palier : elle coupe une pénalité secondaire de moins (triangle)*

Le processus qui permet d'obtenir une fonction de pénalité p_{conv} strictement convexe est le suivant : si lors d'un pas unitaire Δx, la fonction varie de q pas Δy, la coupure coupe seulement $q - 1$ arcs principaux (ou aucun si $q \leq 1$) et q arcs secondaires car elle peut « passer au milieu des arcs de coût mineurs » (voir la figure 3 à droite). On obtient donc la fonction de pénalité p_{conv} de la figure 3 au milieu. On notera que l'on coupe toujours les arcs de coût mineur par paire, ce qui rétablit la fonction de coût initiale.

On a finalement construit un graphe qui permet de calculer l'énergie :

$$\mathcal{E}^d_{conv}(f) = \sum_{i=1}^{n_x} c(x_i, f(x_i))\Delta x + \sum_{i=1}^{n_x - 1} p_{conv}\left(|\Delta f(x_i)|\right) \qquad [9]$$

Grâce à $\mathcal{E}^d_{conv}$, les variations progressives sont maintenant favorisées par rapport aux paliers. Il faut néanmoins remarquer les points suivants.

– Avec le graphe tel qu'il est présenté, il peut être moins pénalisant pour une coupure de couper deux arcs de coût mineurs que de couper un arc de pénalité secondaire. Pour éviter ce problème, on peut soit ajouter une constante à la fonction de coût, ce qui présente les mêmes restrictions qu'une constante sur la fonction de pénalité, soit utiliser les *arcs de contrainte* proposés par Ishikawa [ISH 00] qui éliminent cette constante.

– L'énergie $\mathcal{E}^d_{conv}$ n'a pas directement d'équivalent en continu car p_{conv} dépend du pas de discrétisation. Toutefois, si $\beta = 0$, $\mathcal{E}^d_{conv} = \mathcal{E}^d$ et $\mathcal{E}^d_{conv}$ correspond alors à $\mathcal{E}$. On peut donc avoir une approximation discrète de $\mathcal{E}$ qui favorise les variations progressives aussi bonne que l'on souhaite en prenant β aussi petit que nécessaire.

4.3. *Étude en trois dimensions*

Pour le cas en trois dimensions, on reprend un cadre aussi proche que possible que l'étude 2D : on cherche une fonction f définie sur un rectangle $[a_x, b_x] \times [a_y, b_y]$, qui prend ses valeurs dans $[c, d]$ et qui minimise l'énergie :

$$\mathcal{E}(f) = \int_{a_x}^{b_x}\int_{a_y}^{b_y} \left(c(x, y, f(x, y)) + \alpha_x(x, y) \left| \frac{\partial f}{\partial x}(x, y) \right| + \alpha_y(x, y) \left| \frac{\partial f}{\partial y}(x, y) \right| \right) dx\, dy$$

Nous proposons un procédé totalement équivalent au cas en deux dimensions, nous utilisons simplement une grille en trois dimensions comme base du graphe. Les restrictions à un plan $x = C_x$ ou $y = C_y$ (avec C_x et C_y constantes) sont simplement des graphes à deux dimensions tels que présentés précédemment. Pour une pénalité linéaire, on obtient par conséquent l'élément de base de la figure 4 à gauche. Cette pénalité donne toujours des paliers dans les résultats, on préfère utiliser une approximation par une pénalité strictement convexe, ce qui conduit à l'élément décrit dans la figure 4 à droite. On note simplement que maintenant les arcs de coût mineurs sont coupés par groupes de quatre, ce qui implique qu'ils aient une capacité $\frac{1}{4}c(x, y, z)\Delta x \Delta y$.

Il est important de noter les deux points suivants qui n'apparaissaient pas dans les études précédentes :

– le procédé est anisotrope car il dépend des axes x et y,

– le problème que l'on résout n'a plus de solution avec des techniques de programmation dynamique. La discussion n'entre pas dans le cadre de cet article, l'idée sous-jacente étant que la fonction est définie sur deux variables et que cela empêche un parcours récursif de l'espace des fonctions solutions.

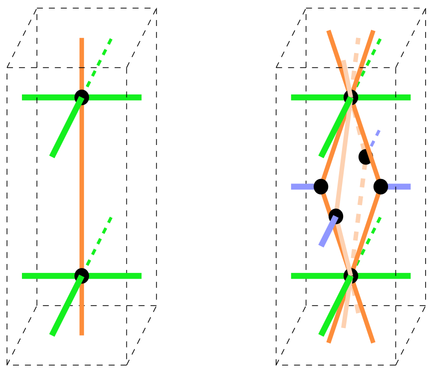

Figure 4. *Gauche : l'élément de base utilisé pour une pénalité linéaire. Droite : pour une pénalité convexe*

4.4. *Extensions*

La méthode présentée permet quelques extensions utiles que l'on présente mais pour lesquelles on ne donne pas de détails techniques par souci de concision.

D + 1 dimensions : De façon totalement similaire, on peut trouver des fonctions à D variables minimisant des énergies dans un espace à $D+1$ dimensions.

$$\mathcal{E}(f) = \int \left(c(X, f(X)) + \sum_{i=1}^{D} \alpha_{x_i}(X) \left| \frac{\partial f}{\partial x_i}(X) \right| \right) dX$$

$$\text{avec} \quad X = (x_1, x_2, ..., x_D)$$

Fonctions périodiques : On peut traiter des fonctions périodiques (selon l'axe x par exemple) simplement en ajoutant des arcs de pénalité entre les nœuds d'abscisse n_x et ceux d'abscisse 1.

Différentes mesures : On peut utiliser d'autres mesures. Pour une fonction définie en coordonnées cylindriques (r, θ, z) par exemple, l'élément d'intégration est $r\, dr\, d\theta\, dz$, on peut distribuer le r sur les fonctions de coût et de pénalité et se ramener au cas d'une fonction périodique en θ.

Approximation locale : L'algorithme de résolution de flot n'étant pas linéaire par rapport au nombre de nœuds de la grille, il peut être utile d'avoir une approximation linéaire du résultat. Pour cela, pour chaque point X où est définie la fonction f on résout le problème d'optimisation sur un voisinage de X et on associe à $f(X)$ la valeur ainsi trouvée. On ne donnera pas ici l'étude complète de cette approximation, on notera simplement que sur des cas pratiques de taille moyenne, les temps de calcul sont similaires au calcul exact. On réservera donc cette approximation au problème de taille importante.

5. Application à la reconstruction 3D

On se propose d'appliquer la méthode d'optimisation proposée dans le cadre de l'algorithme de reconstruction 3D décrit dans [PAR 02]. On ne décrira pas l'ensemble de l'algorithme mais uniquement le cadre dans lequel nous utilisons le flot de graphe.

5.1. *Contexte d'utilisation*

La reconstruction que nous proposons travaille à partir d'une courte séquence d'images très proches les unes des autres. Comme la séquence est courte, l'angle de vue sur les objets à reconstruire est très faible et induit une forte incertitude sur la profondeur des points reconstruits. Comme la méthode d'optimisation à base de graphes trouve le minimum global, nous arrivons à lever cette incertitude.

En pratique, nous travaillons dans un ensemble de voxels qui constitue la discrétisation de l'espace nécessaire à notre méthode. À chaque voxel (x, y, z), nous avons associé une valeur $V(x, y, z)$ qui est d'autant plus faible que le voxel est vu de manière similaire dans toutes les images. Dans l'hypothèse d'objets lambertiens, nous recherchons les faibles valeurs de V car la couleur d'un objet ne dépend pas du point de vue. En pratique, V est la variance des couleurs vues par les caméras : une faible variance indique que le voxel est vu de la même couleur dans toutes les images, ce qui correspond à l'hypothèse lambertienne. Plus de détails sont donnés dans [KUT 99].

Toutefois les images d'entrée n'ayant pas nécessairement une qualité parfaite et à cause de l'ambiguïté sur la profondeur déjà évoquée, nous ne pouvons pas prendre comme surface de l'objet la surface qui minimise uniquement V car elle risque d'être fortement discontinue. Il faut introduire un terme de régularisation et ainsi obtenir une surface qui à la fois est cohérente par rapport aux images et possède une certaine continuité. Nous sommes donc dans le cadre de l'optimisation présentée.

Pour finir, nous n'imposons pas une contrainte de continuité sur l'ensemble de la surface reconstruite car l'objet reconstruit peut très bien présenter des discontinuités. Grâce à l'éclairage, nous savons que ces discontinuités engendrent des variations d'ombrage donc des variations de couleur dans les images. Nous relâchons donc les contraintes de continuité le long des lignes de changement de couleur. Le processus d'optimisation est alors libre de créer ou non une discontinuité le long de ces lignes.

5.2. *Application de la méthode*

Tout d'abord, l'utilisateur fixe une pénalité de discontinuité α_{max} qui doit être appliquée sur les zones de couleur uniforme. Ensuite, à partir des images initiales, nous détectons les discontinuités de couleur horizontales et verticales et construisons ainsi les fonctions de pénalité α_x et α_y (figure 5) qui évoluent entre 0 (sur les fortes discontinuités de couleur) et α_{max} (dans les zones de couleur uniforme). Munis de ces deux fonctions, nous pouvons adapter l'énergie (7) à notre besoin, avec V comme fonction de coût et $\mathcal{D}$ le support de la fonction recherchée :

$$\mathcal{E}(f) = \iint_{\mathcal{D}} \left(V(x,y,f(x,y)) + \alpha_x(x,y) \left| \frac{\partial f}{\partial x}(x,y) \right| + \alpha_y(x,y) \left| \frac{\partial f}{\partial y}(x,y) \right| \right) dx\, dy$$

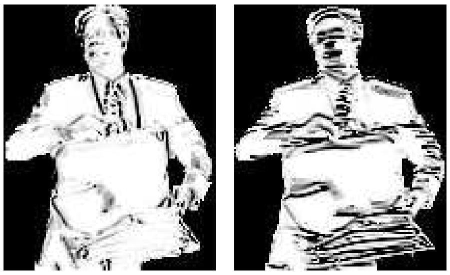

Figure 5. *Exemple de cartes de discontinuité utilisées (gauche : α_x, droite : α_y)*

5.3. *Améliorations obtenues*

Sans cette méthode qui permet d'optimiser la position de la surface dans son ensemble, nous étions obligés de procéder à une optimisation « ligne par ligne » comme beaucoup d'autres méthodes [OHT 85, OKU 93, INT 94, KOC 98, ULV 98]. Cela nous obligeait à avoir de forts traitements dans la suite du processus pour assurer une régularité d'une ligne à l'autre. Ceci avait pour conséquence de « gommer » de nombreux détails de la surface de l'objet (le nez sur un visage par exemple).

Maintenant les traitements qui suivent sont beaucoup plus restreints : on se contente de former une surface à partir de la fonction solution et de la lisser légèrement pour supprimer l'aliassage dû à la discrétisation. Les petits détails sont ainsi conservés.

De plus, le « découpage » en ligne introduisait arbitrairement une séparation entre les directions x et y qui subissaient des traitements différents au cours du processus de reconstruction. Maintenant, ces deux directions sont traitées de manière équivalente, ce qui est plus satisfaisant.

6. Résultats

La figure 6 illustre les résultats obtenus sur trois séquences d'images différentes. Pour juger de ces résultats, il est important de noter la résolution des images utilisées. Par exemple, le visage de l'homme ne couvre dans chaque image qu'une surface d'environ 40×40 pixels. Malgré cela, on remarquera les nombreux détails qui apparaissent dans les modèles géométriques : les touches du clavier, les plis du lampion, le nez de l'homme, le contenu de sa mallette, etc.

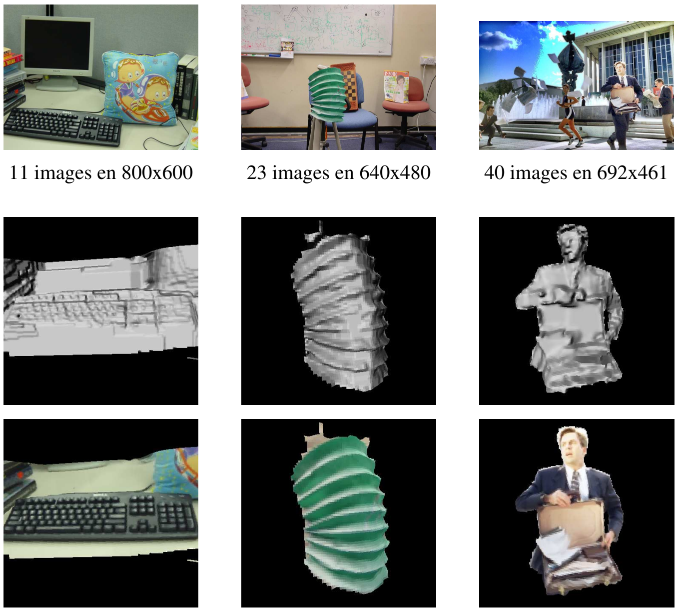

Figure 6. *Résultats sur trois séquences d'images. La première ligne présente une image, la résolution et la longueur de la séquence, la deuxième montre le modèle obtenu sans texture et la troisième le modèle avec texture*

La phase d'optimisation pour le calcul de ces modèles a nécessité de 30 minutes à presque 2 heures selon les modèles sur un processeur MIPS R12000 à 400 MHz et un espace mémoire de 300 méga-octets à 700 méga-octets. Ces chiffres qui peuvent paraître importants sont à relativiser sachant que l'optimisation se fait sur un espace contenant de 1 à 10 millions de voxels et que chaque voxel introduit dans le graphe 5 sommets et 24 arcs (voir la figure 4 à gauche). De plus, en remarquant que le nombre d'arcs est proportionnel à celui de sommets, le calcul de la coupure minimale est en $\mathcal{O}(n^{2,5})$ [CHE 97]. Bien qu'il soit souvent difficile de connaître la taille des graphes utilisés, à notre connaissance, les implémentations actuelles ne dépassent pas l'ordre de grandeur de 100 000 sommets et 300 000 arcs [KOL 02a].

Implémentation

Pour traiter de tels graphes, nous avons apporté une attention particulière à la gestion de la mémoire :

– les relations de voisinage entre sommets et arcs ne sont pas stockées mais calculées à chaque requête,

– seules les fonctions c, α et β sont stockées, ce qui évite de stocker la capacité de chaque arc,

– un arc double (s_1, s_2) permet normalement au flot de « tourner » : il peut exister un flot non nul de s_1 vers s_2 et de s_2 vers s_1, en supprimant la partie « tournante » de ce flot, on peut stocker les flots des deux arcs qui composent (s_1, s_2) sur une seule variable, le signe indiquant la direction du flot.

Nous avons aussi implémenté les heuristiques proposées par Cherkassky et Goldberg [CHE 97] ce qui nous a permis d'atteindre des temps de calcul inférieurs à 24 heures.

7. Conclusion

Nous venons de décrire une nouvelle technique d'optimisation qui s'applique à un ensemble d'énergies que nous avons caractérisé. Cette technique apporte les contributions suivantes :

– une formulation continue du problème à résoudre exhibant le rôle de la géométrie de la configuration,

– la résolution exacte d'une discrétisation de ce problème,

– la mise en évidence d'ambiguïtés ainsi qu'une technique pour les résoudre,

– un graphe original pour représenter certaines fonctions de pénalité convexes,

– l'application de ces résultats à problème d'acquisition 3D à partir d'une courte séquence d'images grâce à une implémentation supportant des graphes de taille très importante.

Grâce à cette méthode nous avons montré que nous pouvons construire des modèles tridimensionnels avec une précision supérieure à celle offerte classiquement par les cartes de disparité et sur des modèles qui nécessitent un volume important de données.

Travaux futurs

Nous pensons que la méthode peut encore être améliorée pour encore mieux reconstruire les surfaces courbes. Nous souhaitons aussi explorer plus précisément les extensions que nous proposons dans la paragraphe 4.4.

Remerciements

Nous tenons à remercier Long Quan et Maxime Lhuillier pour avoir calibré certaines des séquences d'images nécessaires à la validation de la méthode ainsi que Guy Le Besnerais pour son éclairage sur la bibliographie et les discussions qui s'en sont suivies.

8. Bibliographie

[BEL 57] BELLMAN R., *Dynamic Programming*, Princeton University Press, 1957.

[BUE 02] BUEHLER C., GORTLER S., COHEN M., MCMILLAN L., « Minimal Surfaces for Stereo », *European Conference on Computer Vision*, 2002.

[CHE 97] CHERKASSKY B., GOLDBERG A., « On Implementing the Push-Relabel Method for the Maximum Flow Problem », *Algorithmica*, vol. 19, n° 4, 1997, p. 390-410.

[COR 90] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C., *Introduction to Algorithms*, MIT Press and McGraw-Hill, 1990.

[FAU 98] FAUGERAS O., KERIVEN R., « Variational Principles, Surface Evolution, PDE's, Level Set Methods and the Stereo Problem », *IEEE Transactions on Image Processing*, 1998.

[FAU 01] FAUGERAS O., LUONG Q., PAPADOPOULO T., *The Geometry of Multiple Images*, MIT Press, 2001.

[FOR 62] FORD L., FULKERSON D., *Flows in Networks*, Princeton University Press, 1962.

[GOL 97] GOLDBERG A., RAO S., « Length functions for flow computations », rapport n° 97-055, 1997, NEC Research Institute, Inc.

[HAR 00] HARTLEY R., ZISSERMAN A., *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.

[INT 94] INTILLE S., BOBICK A., « Disparity-Space Images and Large Occlusion Stereo », *European Conference on Computer Vision*, 1994.

[ISH 00] ISHIKAWA H., « Global Optimization Using Embedded Graphs », PhD thesis, New York University, 2000.

[KOC 98] KOCH R., POLLEFEYS M., VAN GOOL L., « Multi viewpoint stereo from uncalibrated video sequences », *Lecture Notes in Computer Science*, vol. 1406, 1998.

[KOL 01] KOLMOGOROV V., ZABIH R., « Computing Visual Correspondence with Occlusions using Graph Cuts », *International Conference on Computer Vision*, 2001.

[KOL 02a] KOLMOGOROV V., ZABIH R., « Multi-camera Scene Reconstruction via Graph Cuts », *European Conference on Computer Vision*, 2002.

[KOL 02b] KOLMOGOROV V., ZABIH R., « What Energy Functions can be Minimized via Graph Cuts ? », *European Conference on Computer Vision*, 2002.

[KUT 99] KUTULAKOS K., SEITZ S., « A Theory of Shape by Space Carving », *International Conference on Computer Vision*, 1999.

[OHT 85] OHTA Y., KANADE T., « Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, 1985, p. 139–154.

[OKU 93] OKUTOMI M., KANADE T., « A Multiple-Baseline Stereo », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, n° 4, 1993, p. 353–63.

[PAR 02] PARIS S., SILLION F., « Robust Acquisition of 3D Informations from Short Image Sequences », *Pacific Graphics*, 2002.

[ROY 98] ROY S., COX I., « A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem », *International Conference on Computer Vision*, 1998.

[SLA 01] SLABAUGH G., CULBERTSON B., MALZBENDER T., SCHAFER R., « A Survey of Methods for Volumetric Scene Reconstruction from Photographs », *VolumeGraphics*, 2001.

[ULV 98] ULVKLO M., KNUTSSON H., GRANLUND G., « Depth segmentation and occluded scene reconstruction using ego-motion », *Proc. SPIE Vol. 3387, p. 112-123, Visual Information Processing VII*, 1998.

[VEK 99] VEKSLER O., « Efficient Graph-Based Energy Minimization Methods in Computer Vision », PhD thesis, Cornell University, 1999.

Sylvain Paris *est étudiant en thèse sous la direction de François Sillion au sein de l'équipe ARTIS à Grenoble. Ses travaux portent sur l'acquisition, la modélisation et le rendu d'informations tridimimensionnelles à partir d'images.*

François Sillion *est Directeur de Recherche à l'INRIA, l'Institut National de Recherche en Informatique et en Automatique, responsable scientifique du projet de recherche ARTIS à Grenoble. Il est également enseignant à temps partiel à l'École polytechnique. Ses centres d'intérêt sont la simulation de l'éclairage en image de synthèse, la visualisation interactive de très gros volumes de données, les techniques de rendu non-photoréaliste et le rendu inverse.*