Projet de Reconstruction 3D

A Hand-help Photometric Stereo Camera for 3D Modeling

Vincent Delaitre

vincent.delaitre@ens-lyon.fr

1 Introduction

Cet article propose une nouvelle approche pour la reconstruction 3D basée sur le caractère non Lambertien des matériaux réels. La méthode proposée permet de retrouver non seulement la normale et la profondeur mais aussi l'albédo et la lumière ambiante à partir d'un ensemble de photos. Les contraintes imposées par les précédents algorithmes (caméra orthographique, lumière éloignée ou de position connue, caractère Lambertien des objets étudiés) ne sont plus utiles et seule subsiste la nécessité d'une source de lumière proche de l'objet et fixe dans le référentiel de la caméra.

L'algorithme proposé se décompose en quatre étapes. Les deux premières utilisent des méthodes déjà existantes pour la calibration des paramètres intrinsèques à l'appareil photo et le calcul des matrices de projection pour chaque prise de vue [3].

Les deux dernières étapes relèvent de la contribution des auteurs. Ils définissent une énergie à minimiser et calculent par graph-cut la profondeur des points de l'objet. Cette solution approchée est ensuite raffinée dans la dernière étape.

2 Algorithme proposé

2.1 Modèle photométrique en lumière-proche

On suppose que l'on a m images, dont les matrices de projection (rotation $\boldsymbol{R}_i$ et translation $\boldsymbol{t}_i$ pour $i \in \{1, \cdots, m\}$) sont données par les deux premières étapes de l'algorithme. On suppose aussi connue la position $\boldsymbol{s}$ de la source de lumière, fixe dans le référentiel de l'appareil photo.

Alors, étant donné un point $\boldsymbol{x}$ de l'objet exprimé dans les coordonnées de la scène, on peut calculer le vecteur lumière $\boldsymbol{l_i}$ allant du point à la source dans le référentiel de la caméra :

$$\boldsymbol{l_i} = \boldsymbol{s} - (\boldsymbol{R}_i \boldsymbol{x} + \boldsymbol{t}_i) \tag{1}$$

En posant $\boldsymbol{l'_i} = \boldsymbol{R_i}^T \boldsymbol{l_i} / |\boldsymbol{l_i}|^3$, on peut relier o'_i l'intensité de lumière observée en $\boldsymbol{x}$ à la quantité de lumière ambiante a', l'albédo de l'objet ρ ainsi qu'à la position du vecteur lumière par rapport à la normale $\boldsymbol{n}$ en $\boldsymbol{x}$:

$$o'_i = \boldsymbol{l'_i} \cdot (\rho \boldsymbol{n}) + a' \tag{2}$$

Chaque point de l'espace génère donc quatre inconnues (trois pour le vecteur $\rho\boldsymbol{n}$ et une pour a') qui peuvent être déterminées à partir du moment où l'on sait repérer ce point sur quatre vues différentes.

2.2 Calcul de la profondeur

Les auteurs utilisent la stéréographie dite "plane-sweep" : ils choisissent une vue de référence et projettent les autres vues dans des plans de profondeur variable et discrétisée.

Ils calculent alors les caractéristiques des points de cette vue de référence (albédo, normale et lumière ambiante) en utilisant RANSAC pour maximiser le nombre de vues qui s'accordent avec ces caractéristiques.

Un graph-cut permet alors de calculer la surface optimale. Pour cela ils définissent une énergie composée de trois termes. On calcule ainsi pour chaque pixel l'énergie de cohérence à la photo E_p

qui dépend du nombre de vues en accord avec les caractéristiques calculés, l'énergie de contrainte normale E_n qui impose que la surface soit en accord avec les normales déterminées et l'énergie de contrainte de profondeur E_s qui empêche une trop grande discontinuité de la profondeur.

Cependant, la surface calculé par le graph-cut est néanmoins assez grossière à cause de la discrétisation de la profondeur. Les auteurs raffinent le résultat en la faisant évoluer de manière à ce qu'elle minimise une énergie imposant un certain lissage et un respect des normales calculées auparavant sans toutefois permettre de s'éloigner trop de la solution discrétisée.

3 Réalisations

J'ai réalisé la majeure partie des algorithmes proposés dans l'article et j'en détaille l'implémentation dans cette section.

3.1 Plane-sweep stéréo

Pour réaliser une plane-sweep stéréo, il faut commencer par connaitre la normale aux plans balayés. Pour cela j'estime le centre de la scène en me servant des informations sur la position des caméras délivrées par Bundler [3]. Malheureusement, certaines positions sont parfois erronées. J'utilise donc un RANSAC : pour deux caméras données, je définis un centre approximatif $\boldsymbol{O}$ comme étant le point équidistant à leurs axes. Les inliers sont les caméras telles que la projection du point O ne soit pas trop éloignée du centre de l'image.

Une fois le centre de la scène connu, je calcule la normale K en sommant les vecteurs $\boldsymbol{OC_i}$ (avec $\boldsymbol{O}$ le centre de la scène et $\boldsymbol{C_i}$ le centre de la i^{ème} caméra). Je munis alors l'espace d'un repère orthonormal $(\boldsymbol{I}, \boldsymbol{J}, \boldsymbol{K})$. Dans la suite, les coordonnées (x, y, z) des points de l'espace seront exprimées dans ce repère.

Il faut alors limiter l'espace de recherche, c'est-à-dire en trouver les bornes pour chaque dimension. Si on se place dans un plan de profondeur z donnée, on peut facilement calculer les polygones (éventuellement non fermés) correspondant à la projection des images sur ce plan. De plus, si ce plan intersecte la scène alors ces polygones vont se recouvrir en grande partie.

Notons c le rapport de la surface de l'intersection de tous ces polygones sur la surface du plus petit polygone. Je détermine z_{min} et z_{max} en balayant la profondeur à partir de $z_0 =< \boldsymbol{O}|\boldsymbol{K} >$ et en m'arrêtant lorsque c passe en dessous d'un certain seuil. Pour borner x et y, on choisit simplement le plus petit rectangle qui englobe l'intersection des polygones pour $z_{min} \leq z \leq z_{max}$.

3.2 Détermination du facteur d'échelle

La détermination de la position des caméras simplement à partir des images est définie à homothétie près. Par conséquent, le vecteur lumière définit section 2.1 s'écrit en fait :

$$\boldsymbol{l_i} = \boldsymbol{s} - k(\boldsymbol{R}_i\boldsymbol{x} + \boldsymbol{t}_i) \tag{3}$$

avec k le facteur d'échelle. Pour déterminer k, on utilise les points caractéristiques fournis par Bundler : on calcule normales, albédo et lumière ambiante en ces points pour différentes valeur de k et on choisit le k qui minimise l'énergie d'attache aux données E_p.

3.3 Calcul de la surface par graph-cut

Les énergies E_p et E_n sont définies dans l'article mais comportent quelques imprécisions. Je détaille ici les ambiguïtés rencontrées et les solutions adoptées.

Énergie d'attache aux données $\boldsymbol{E_p}$
On utilise RANSAC pour calculer les caractéristiques de chaque pixel de l'espace de recherche. Les auteurs calculent le résidu du RANSAC en utilisant les trois composantes couleurs pour l'albédo et la lumière ambiante (cf équation 6 de l'article). Néanmoins, leur système à quatre équations utilisé section **??** ne permet que de déterminer l'albedo ρ et la lumière ambiante a en niveau de gris et ils ne précisent pas comment obtenir ρ^c et a^c pour $c \in \{R, G, B\}$.

Ma démarche est donc la suivante. En notant E^c l'intensité de lumière émise par l'appareil photo à distance unité, on a pour chaque couleur $c \in \{R, G, B\}$:

$$o_i^c = E^c \rho^c (\boldsymbol{l'_i} \cdot \boldsymbol{n}) + a^c \tag{4}$$

En posant $\rho'^c = \rho^c E^c$ et en faisant l'hypothèse que $a^c = \rho'^c a$, on peut alors écrire que :

$$\rho'^c = \frac{o_i^c}{(\boldsymbol{l'_i} \cdot \boldsymbol{n} + a)} \tag{5}$$

On calcule donc $\boldsymbol{n}$ et a à partir du système d'équation en niveau de gris, on en déduit ρ'^c puis a^c. Le reste ne présente pas de difficulté.

Énergie de régularisation tangentielle $\boldsymbol{E_n}$
Cette énergie revient à favoriser le fait que les normales $\boldsymbol{n}$ calculées précédemment soient perpendiculaires à la surface : l'idée est d'associer à chaque point $\boldsymbol{p}$ une pénalité du type $\boldsymbol{T} \cdot \boldsymbol{n}$ avec $\boldsymbol{T}$ le vecteur tangent à la surface en $\boldsymbol{p}$.

L'énergie de l'article fait intervenir une somme et une constante T_j qui ne sont pas clairement définit. Je procède donc comme suit. La normale en un pixel 2D $\boldsymbol{p}$ du plan de profondeur z_j (j^{ième} plan pour la plane-sweep stéréo) est notée $\boldsymbol{n_p^j}$. Toujours dans le plan, j'appelle d la distance entre $\boldsymbol{p}$ et un voisin diagonal et je pose $T_j = Td$ avec T une constante (on remarque que T_j est une constante indépendante de j, ce qui laisse penser que je n'ai pas la bonne définition de T_j).

Ensuite, pour tout pixel $\boldsymbol{p}$ sur le plan de profondeur z_j et pour chacun des huit voisins $\boldsymbol{p'}$ de $\boldsymbol{p}$, je cherche la profondeur j' minimise la distance du point 3D $(p'_x, p'_y, z_{j'})$ au plan normal à $\boldsymbol{n_p^j}$. Si RANSAC a pu estimer une normale $\boldsymbol{n_{p'}^{j'}}$, je note $\boldsymbol{d_{p\ j}^{p'j'}}$ le vecteur 3D de $(\boldsymbol{p}, z_j)$ à $(\boldsymbol{p'}, z'_j)$, sinon je cherche le second meilleur j', etc... Je pose alors :

$$g(p, j, p') = \begin{cases} |\boldsymbol{d_{p\ j}^{p'j'}} \cdot \boldsymbol{n_{p'}^{j'}}|(|j - j'| + 1) \text{ si } \exists j' \text{ tq } \boldsymbol{n_{p'}^{j'}} \text{ existe et } |j - j'| \leq T_j \\ C_0/8 \qquad\qquad\qquad\qquad \text{sinon} \end{cases} \tag{6}$$

Je défini alors E_n par :

$$E_p(p, j) = \sum_{\boldsymbol{p'} \text{ voisin 2D de } \boldsymbol{p}} g(p, j, p') \tag{7}$$

Graph-cut
La construction du graph-cut ne pose pas de problème. On cherche à minimiser l'énergie :

$$E = E_p + \lambda_n E_n + \lambda_s E_s \tag{8}$$

avec $E_s(j, j') = \Delta_z |j - j'|$ (en notant Δ_z l'écart de profondeur entre deux plans de la plane sweep). On construit donc un graphe à plusieurs niveaux, un pour chaque profondeur. Les arêtes entre niveaux sont étiquetées par $E_p + \lambda_n E_n$. Pour prendre en compte l'énergie E_s, on attribue le poids $\lambda_s \Delta_z$ aux arêtes entre voisins sur un même niveau.

4 Affinage de la surface

Afin d'affiner la surface obtenue par graph-cut, les auteurs proposent une nouvelle énergie composée d'un terme d'attache à la surface initial E^P, d'un terme incitant les normales calculées à être réellement les normales de la surface E^N et d'un terme limitant le Laplacien.

Cette énergie va être minimisée globalement par la technique des moindres carrés. Sans rentrer dans les détails, chaque pixel va générer quatre équations et conduire au système suivant (je l'ai modifié car il me semble que le système présenté dans l'article ne minimise pas tout à fait l'énergie voulue) :

$$\begin{bmatrix} \sqrt{\lambda_1}\mu \\ \sqrt{1-\lambda_1}\mathcal{N}\cdot\mathcal{T}^x \\ \sqrt{1-\lambda_1}\mathcal{N}\cdot\mathcal{T}^y \\ \sqrt{\lambda_2}\nabla^2 \end{bmatrix} [Z] = \begin{bmatrix} \sqrt{\lambda_1}\mu Z^* \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{9}$$

avec Z la surface recherchée, Z^* la surface initiale, avec μ une matrice diagonale ayant μ_i comme i^{ième} coefficient (défini dans l'article), ∇^2 la matrice calculant la contrainte sur le laplacien et $\mathcal{N}\cdot\mathcal{T}^x$ et $\mathcal{N}\cdot\mathcal{T}^y$ les matrices exprimant les contraintes tangentielles en x et y.

La matrice du système possède un grand nombre de coefficients ($4 * N^2$ avec N le nombre de pixels) mais seulement au plus 9 coefficients par ligne sont non nuls. J'ai donc implémenté une classe C++ particulière pour manipuler de telles matrices, ainsi qu'un algorithme de minimisation par moindre carré pour calculer la surface raffinée.

4.1 Approche pyramidale

L'ensemble des opérations ci-dessus est répété dans ce que j'appelle une 'itération'. Entre deux itérations, on divise par 2 la distance Δ_z entre deux plans de balayage voisins. Pour éviter une explosion des calculs lors de l'itération suivante, on restreint le balayage de j_p aux profondeurs comprises entre le minimum moins 1 et le maximum plus 1 des profondeurs des voisins de $\boldsymbol{p}$.

5 Utilisation du programme

5.1 Installation de Bundler

Une version binaire de bundler est fournie dans le répertoire "projet/bundler". Autrement, il peut être téléchargé à l'adresse suivante : `http://phototour.cs.washington.edu/bundler/`.

Bundler requiert l'installation des paquets LAPACK, BLAS, CBLAS, MINPACK, f2c (cf "projet/bundler/INSTALL").

De plus, il faut modifier la ligne 16 du fichier "projet/bundler/RunBundler.sh" et remplacer "TODO" par le chemin absolu du répertoire d'installation.

5.2 Utilisation

La marche à suivre est la suivante :

- Se placer dans le répertoire contenant les images à traiter.
- Lancer RunBundler.sh (exemple à partir de "img/kermit" : "../../projet/bundler/RunBundler.sh"). Pour les exemples fournis, il est possible d'utiliser le makefile : "make bundler".
- Lancer PhotoStereo (exemple à partir de "img/kermit" : "../../projet/src/PhotoStereo"). Pour les exemples fournis, il est possible d'utiliser le makefile : "make stereo".
- Le programme affiche en sortie la profondeur avant et après raffinage, les normales, l'albédo, la lumière ambiante ainsi que le maillage 3D.

5.3 Options du programme

La syntaxe est la suivante :

```
./PhotoStereo [-v] [-p path] [-n nb_iter]
```

- `-v` : affiche des informations de débuggage. (désactivé par défaut)
- `-n nb_iter` : règle le nombre d'itérations à nb_iter : augmenter le nombre d'itérations augmente la précision mais aussi le temps de calcul. (min = 1, max = 4, défaut = 2)
- `-p path` : indique le chemin d'accès vers le répertoire contenant les images. (défaut = './')

D'autre part, les répertoires fournis en exemple contiennent un fichier params.ini regroupant les paramètres possibles. Si ce fichier n'existe pas, les valeurs par défaut sont utilisées.

6 Résultats

J'ai contacté l'auteur de l'article et j'ai pu me procurer un des jeux d'images tests utilisés dans l'article : la statue de marbre. Malheureusement les résultats expérimentaux ne sont pas exceptionnels comme on va le voir ci-dessous.

N'ayant pas de base line, je ne ferai que présenter des résultats qualitatifs. Dans la suite, il arrivera que j'exhibe une carte des normales. Celles-ci seront représentées par des couleurs selon la carte figure 1.

Fig. 1. Code couleur des normales (chaque point de la boule unité donne sa couleur à la normale correspondante)

Kermit
N'ayant au début qu'à disposition des images utilisées en stéréographie classique, j'ai commencé par tester le programme sur ces images représentant Kermit, la grenouille du Muppet Show. La reconstruction s'effectue de manière grossière (cf figure 2 et 3) malgré le fait que cette scène ne respecte absolument pas le modèle photométrique qui sert de base à l'algorithme. Il y a en effet une source de lumière importante venant de la droite de la scène, ce qui mène à des normales incohérentes.

La statue de marbre
La statue de marbre est un ensemble de 100 photos que je me suis procuré auprès de l'auteur de l'article. Parmi ces photos, Bundler ne parvient à en classer que 31. De nombreuses caméras sont mal positionnées et mon programme n'en sélectionne qu'une partie pour finalement en garder 13. Malheureusement, ces 13 vues sont encore incohérentes et la reconstruction ne même à aucun résultat probant. Même manuellement en balayant l'espace des profondeurs, on observe aucune corrélation entre les images comme c'est d'habitude le cas. (cf figure 4). L'idéal aurait été d'indiquer à la main au programme quelles images sélectionner mais je n'ai pas eu le temps de le faire.

Fig. 2. A gauche : un exemple de quatre images sources pour Kermit. A droite : caractéristiques du modèle 3D, de gauche à droite et de haut en bas : profondeurs, normales, albédos et lumière ambiante

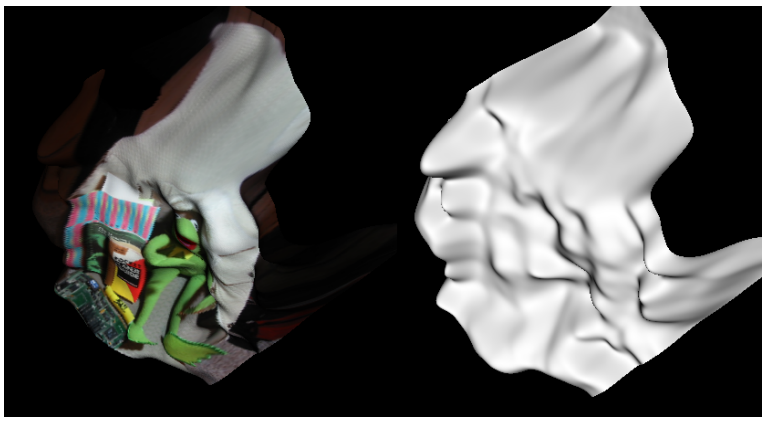

Fig. 3. Reconstruction 3D renvoyée par le programme.

La guitare

J'ai aussi essayé de reconstruire une scène à partir de mes propres photos. J'ai donc pris des vues d'une guitare posée sur une chaise en activant le flash et en tamisant la lumière extérieure. Cette scène comporte au moins deux difficultés pour la reconstruction.

D'une part elle allie objets mats (ex : la chaise) et des objets brillants (ex : la guitare vernie). Des irrégularités de profondeur apparaissent aux endroits où le flash se réfléchit dans la guitare si bien que j'ai dû supprimer les images concernées pour résoudre le problème.

D'autre part, la scène comporte des objets qui occulte ou non certaines zones suivant l'endroit de la prise de vue (les accoudoirs de la chaise cachent le parquet et la guitare masque différents endroits du dossier suivant la photo). Cela perturbe les fonctions d'énergie et des discontinuités de profondeur apparaissent autour de ces endroits (cf figure 5). En effet, la profondeur réelle du point ne reçoit pas un bon score car il est caché dans certaines vues.

Les résultats de reconstruction sont montrés figures 6 et 7.

L'ours en peluche

On pourrait penser qu'un flash d'appareil photo est trop puissant et gomme les différences de luminosité dues au terme faisant intervenir les normales de l'objet. J'ai donc pris des photos d'un ours en peluche en éclairant la scène avec une lampe de poche collée à l'appareil. Malheureusement, elle devait manquer de puissance car la reconstruction obtenue est assez mauvaise et c'est finalement le jeu de photos prisent avec flash qui approximent le mieux l'ours (cf figure 8 et 9).

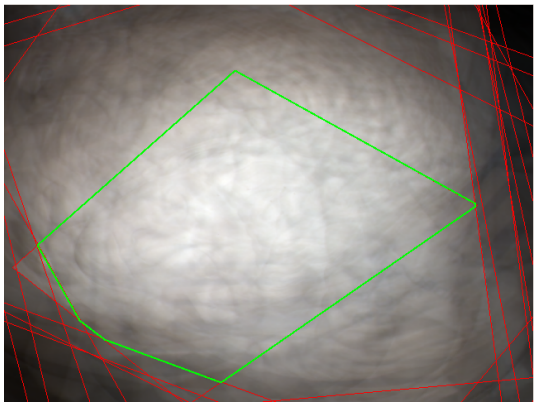

Fig. 4. Un exemple de projection des images de la statue sur un plan censé être incident à l'objet. Quelque soit la profondeur, on n'observe aucune corrélation entre les images.

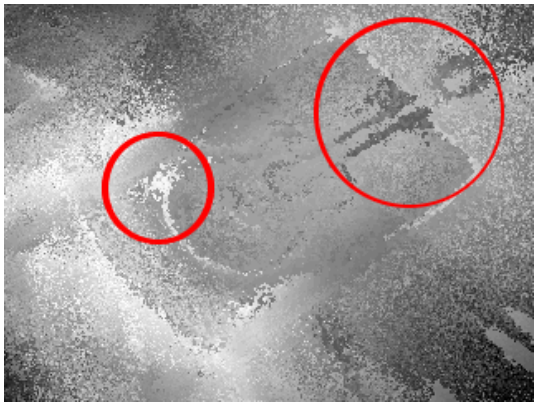

Fig. 5. Discontinuité de la profondeur au niveau des régions occultées dans une partie des vues (petit cercle : certaines caméras voient le parquet à travers l'accoudoir, grand cercle : la partie droite ou gauche du dossier est occultée suivant le côté où l'on se place)

Fig. 6. A gauche : un exemple de quatre images sources. A droite : caractéristiques du modèle 3D, de gauche à droite et de haut en bas : profondeurs, normales, albédos et lumière ambiante

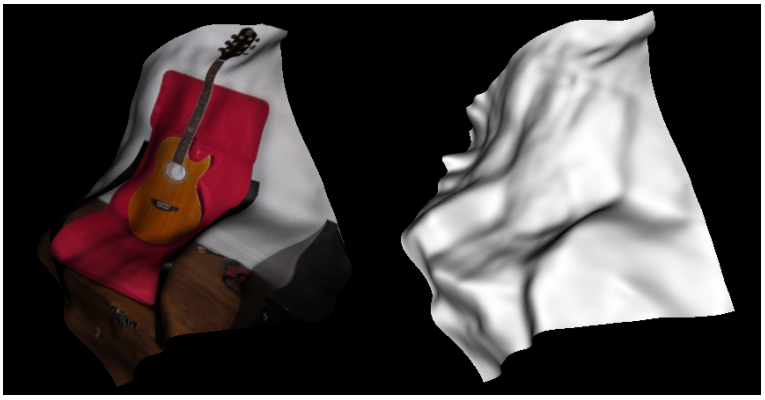

Fig. 7. Reconstruction 3D renvoyée par le programme.

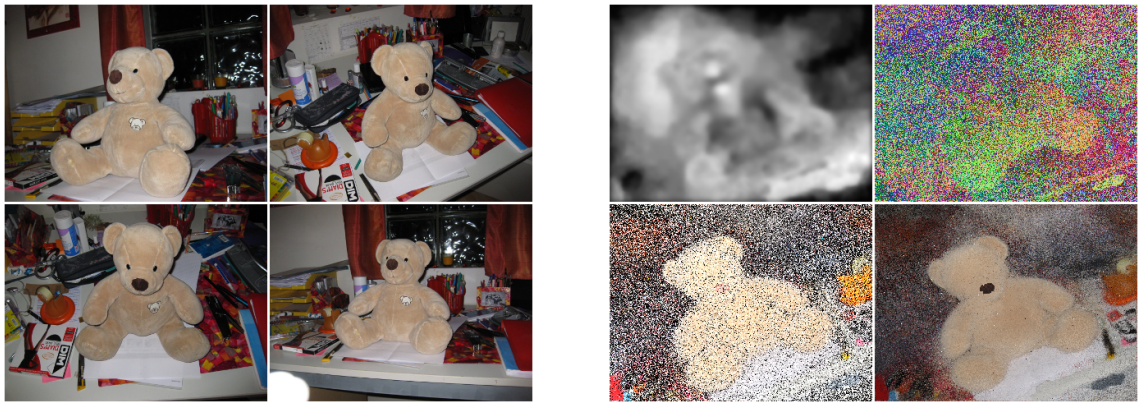

Fig. 8. A gauche : un exemple de quatre images sources. A droite : caractéristiques du modèle 3D, de gauche à droite et de haut en bas : profondeurs, normales, albédos et lumière ambiante

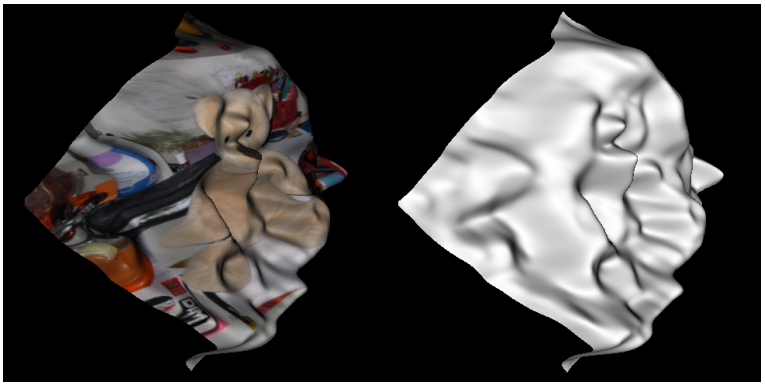

Fig. 9. Reconstruction 3D renvoyée par le programme.

7 Conclusion

L'article se présente comme un article assez novateur dans la mesure où il est possible de s'affranchir de nombreuses contraintes requises usuellement. En comparaison avec un algorithme de stéréographie classique, il permet l'obtention d'informations supplémentaires telle que l'albédo et la lumière ambiante. Bien que les résultats de précision ne soient pas vraiment meilleurs que l'algorithme de référence, les reconstructions effectuées sont qualitativement très bonnes.

Néanmoins, certains points de l'article sont passés sous silence, probablement pour des raisons de concision. Je citerai par exemple l'obtention de l'albédo pour les trois canaux de couleur dans E_p ou la démarche exacte pour calculer E_n. Les choix que j'ai effectués pour combler ces zones d'ombres pourraient expliquer la moins bonne qualité de mes reconstructions. Il peut aussi s'agir d'un bug (caché car je n'observe jamais de segmentation fault) ou du fait que les paramètres sont mal réglés (j'ai pourtant passé beaucoup de temps à jouer avec et de plus, les valeurs qui me semblent pertinentes sont bien celles exposées dans l'article).

En plus de tester de nombreuses valeurs des paramètres, j'aurai essayé de nombreuses idées : changer l'importance donnée au facteur N dans la valeur de E_p, différentes manières de calculer la normale, l'albédo et la lumière ambiante en un point (notamment par des techniques de moindre carré) ; augmenter la taille du voisinage à considérer pour l'approche pyramidale et flouter la carte des profondeurs entre deux itérations pour ne pas s'enfermer trop vite dans un intervalle de profondeur erronée ; imposer une énergie nulle au bord pour éviter que l'arrière-plan ne jaillisse en avant ; etc...

Il me reste pourtant une idée que je n'ai pas testée. Il est possible que la réelle différence se situe dans l'affinage de la surface. J'effectue cet affinage dans une caméra virtuelle qui regarde perpendiculairement aux plans de la plane sweep stereo. Je crois cependant comprendre que l'auteur projète la surface discrète dans chaque caméra (en supprimant les parties cachées), l'affine, puis fusionne les modèles obtenus. Malheuresement, il m'aurait fallu plus de temps pour creuser cette idée qui est la suite logique de ce projet.

Références

1. C. Paige and M. Saunders. Lsqr : An algorithm for sparse linear equations and sparse least squares. In *ACM Trans. on Mathematical Software*, page 43–71, 1982.
2. D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi. Efficiently combining positions and normals for precise 3d geometry. In *SIGGRAPH*, page 536–543, 2005.
3. N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism : Exploring photo collection in 3d. In *SIGGRAPH*, pages 835–846, 2006. `http://phototour.cs.washington.edu/bundler/`.
4. T. Higo, Y. Matsushita, N. Joshi, and K. Ikeuchi. A hand-help photometric stereo camera for 3d modeling. In *ICCV*, pages 1234–1241, 2009.