SceneSailing

Navigation 3D sans Reconstruction Explicite

3D Navigation Without Explicit Reconstruction

Julien Peyras[1] Adrien Bartoli[1] Pierre Georgel[2] Marco Loog[3] *

[1] LASMEA, Clermont [2] TU-München [3] University of Delft

prénom.nom@gmail.com

Résumé

Naviguer dans une collection d'images est utile pour de nombreuses applications qui, par soucis de réalisme, font usage du Rendu Basé Image *(Image-Based Rendering). La navigation est cependant tributaire de l'estimation correcte des caméras à l'origine de chacune des vues de la collection. Ce problème d'estimation est souvent résolu par* Structure-from-Motion *(SfM) qui consiste à calculer les caméras pour des images dans une collection, et qui laisse l'utilisateur placer une caméra virtuelle parmi celles-ci. Cette solution présente certains inconvénients en pratique, le plus important étant le cas pour lequel la SfM échoue (dû par exemple au manque de texture ou au mouvement dégénéré de caméra).*

Nous proposons un système qui permet à l'utilisateur de naviguer intuitivement dans une collection d'images sans qu'aucune caméra n'aie besoin d'être calculée explicitement. Nous calculons le mouvement local inter-image que nous fournissons à un algorithme de réduction de dimension. Ceci conduit à un espace des paramètres de mouvement à faible dimension à l'intérieur duquel l'utilisateur peut directement naviguer et explorer la variété d'images. Le système fait usage d'une interface de capture de mouvement qui procure une forte sensation d'immersion à l'utilisateur, même le plus inexpérimenté.

Mots Clef

Rendu Basé Image, Navigation 3D, Vidéo, Interface Homme-Machine (IHM), Interpolation, Morphing, Structure-from-Motion (SfM).

Abstract

Navigating in a collection of images is useful to many applications based on the Image-Based Rendering to ensure the maximum rendering realism. The possibility to navigate depends however on the correct estimate of the cameras that originated the pictures of the collection. This problem is usually solved by means of Structure-from-Motion that gives the cameras for the images in the collection, and lets the user to place the virtual camera amongst those. This solution may have several shortcomings in practice, the most important one being those cases for which Structure-from-Motion fails (due for instance to lack of texture or degenerate camera motion).

We propose a means that allows a user to intuitively navigate in a collection of images without any need for camera estimation. Instead, we compute a local inter-image simple motion model that we feed in a dimensionality reduction like algorithm. This reveals a low dimensional camera parameter space within which the user can directly sail and explore the image manifold. We set up a motion capture interface that makes the user, even unused to the technology, experience a strong immersive feeling.

Keywords

Image-Based Rendering (IBR), Navigation 3D, Vidéo, Human-Machine Interface, Interpolation, Morphing, SfM.

1 Introduction

A l'ère de la visite virtuelle, aussi bien d'une scène synthétique (*e.g.* les jeux vidéos) que réelle (*e.g.* les musées), nous assistons à une demande grandissante d'outils simples permettant de manière rapide et réaliste de se déplacer dans une scène 3D. Dans le cas de la scène synthétique on dispose du modèle de la scène observée. Il est donc simple de spécifier la *caméra virtuelle* définissant la vue souhaitée et de faire le rendu de cette vue. Cependant, malgré les intenses efforts consacrés à obtenir une qualité excellente de rendu, les images synthétiques n'ont jamais atteint le niveau de réalisme qu'offrent les images naturelles. C'est pourquoi le Rendu Basé Image (*Image Based Rendering* (IBR) en anglais) reçoit actuellement une si grande attention [2, 6, 5, 3]. Avec [3], Chen et Williams sont sans doute

*Nous tenons à remercier infiniment l'ANR pour le support financier permettant le bon déroulement de ce projet.

les premiers en 93 à outrepasser les règles régissant alors le rendu visuel. En proposant QuickTimeVR, les auteurs initient la communauté à l'utilisation d'images naturelles pour faire du rendu réaliste.

La plupart des techniques d'IBR se basent sur la fonction plénoptique [1] qui décrit la couleur des rayons lumineux voyageant dans l'environnement. Les images réelles constituent autant d'échantillons de cette fonction. Un concept différent, et que nous utilisons dans ce papier, est celui de la *variété d'images* [4]. Une image est considérée comme étant représenté par un point dans un espace de dimension haute, égale au nombre de pixels de cette image. La collection d'images dont nous disposons forme la variété. Elle est généralement de faible dimension. Sous cette représentation du problème, spécifier la caméra consiste à parcourir de possibles paramétrisations de la variété d'images, alors que faire le rendu consiste à reconstruire, au moins localement, la variété d'images par interpolation des images réelles.

L'utilisation d'images naturelles pose deux problèmes principaux : la spécification de la caméra virtuelle et le rendu de la scène vue par cette caméra. Deux types de solutions existent pour affronter ces problèmes. La première catégorie, connue sous le nom de *Image-Based Modelling*, propose de reconstruire explicitemment la structure 3D de la scène et d'estimer la position de caméra pour chacune des vues disponibles. Snavely *et al.* [2] ont proposé une solution portant le nom de PhotoSynth pour voyager à travers une large quantité d'images représentant la même scène observée depuis plusieurs points de vue. Les caméras sont estimées pour permettre à l'utilisateur de se situer dans l'espace et de changer de vue. Levoy et Hanrahan ont proposé le *Light Field Rendering* [6], une limitation de la fonction plénoptique à 4 dimensions. Le *light field* est estimé moyennant la connaissance exacte des caméras. Le Lumigraph [5] de Gortler *et al.*en est une extension. Les caméras sont calculées grâce à des marqueurs de calibration placés dans la scène. Il n'est cependant pas toujours possible d'inférer correctement la structure de la scène grâce à ces méthodes qui fonctionnent mal en présence de zones non-texturées, d'éléments à la structure et apparence trop complexes ou de mouvements dégénérés de caméra.

La seconde catégorie de solutions ne tente pas de reconstruire la scène explicitement, mais elle en analyse localement la topologie pour inférer de nouvelles vues à partir de celles disponibles. Fusiello a proposé dans [7] le moyen de faire de l'extrapolation de vues depuis des données non-calibrées, en modélisant la trajectoire probable de la caméra. Dans leurs travaux [9] utiles pour des applications basées sur le morphing, Lhuillier et Quan ont proposés plusieurs méthodes de mise en correspondance entre images voisines, suivi de leur *joint view triangulation* visant à rendre non ambigue les zones présentant des sauts de profondeur. Des méthodes hybrides sont donc possibles, basées en partie sur une modèlisation explicite des caméras, et sur une interpolation des images voisines pour faire le rendu.

FIG. 2 – Exemple d'une variété représentée dans les 2 dimensions principales (parmi les 8) de l'espace des paramètres. Quelques données images correspondantes à plusieurs positions sont présentées.

Nous entrons dans une époque où le stockage de données et la vitesse d'accès à celles-ci pose de moins en moins problème. Cette avancée rend désormais possible l'utilisation de l'IBR pour la mise au point d'applications temps réel. Nous proposons une méthode de navigation dans une collection d'images qui fonctionne à partir d'une vidéo dense de la scène réelle que l'on souhaite pouvoir explorer interactivement. Nous suivrons dans ce papier l'exemple d'une séquence complexe dont un extrait est fourni en figure 1.

La redondance d'images permet, en exploitant le voisinage entre données, de nous soustraire d'une contraignante reconstruction explicite. Chaque image possède de nombreuses voisines qui ne diffèrent que par une quantité faible de mouvement dû au déplacement restreint de la caméra. En mesurant ce mouvement local nous caractérisons correctement la répartition relative de toutes les vues disponibles. Les nombreux recoupements de vues dans la vidéo assurent de proche en proche que la répartition des données est finalement correcte de façon globale. De cette manière nous nous affranchissons du problème coûteux et souvent complexe de la caractérisation absolue de la pose des caméras.

Le mouvement entre paires d'images voisines caractérise la topologie locale de la variété observée que l'on peut exprimer dans un *espace des paramètres de mouvement*. Cet espace est de faible dimension : nous utilisons ici l'homographie, définie par 8 paramètres, pour estimer le mouvement entre deux images. En plus d'une réduction de dimension des données, l'espace des paramètres offre une sémantique claire et utile : les données y sont agencées de telle sorte que le mouvement qui sépare les images correspondantes est respecté. Il devient alors intuitif pour un utilisateur de passer d'une vue de la scène à une vue voisine au gré de son envie de bouger dans une direction ou une autre. La figure 2 donne une représentation de l'espace des paramètres et

FIG. 1 – Extrait d'images provenant d'une collection représentant une scène complexe filmée de manière dense.

des données images correspondantes. Pour permettre cette navigation libre à l'utilisateur nous proposons d'interfacer son mouvement par capture du déplacement d'un motif plan manipulé devant une webcam. Quand l'utilisateur lui applique un déplacement, le mouvement est retranscrit à l'identique dans l'espace des paramètres. C'est ainsi que nous résolvons l'épineux problème de la navigation au sein des données.

Afin d'assurer une fluidité majeure de la navigation, la variété est rendue continue par application d'une fonction noyau à chacune des données. On définit ainsi un voisinage continu autorisé autour des données. A tout point de cette variété continue correspond une vue reconstruite de la scène par interpolation des données voisines.

En itérant sur la capture du mouvement de l'utilisateur, sa retranscription dans l'espace des paramètres, et l'affichage de l'image correspondant à la nouvelle position dans cet espace, l'utilisateur fait l'expérience d'un *Sailing* immersif et intuitif. Le fait que les données images soient naturelles contribue évidemment énormément à la sensation d'immersion.

Ce travail fournit ainsi une solution très simple et intuitive pour explorer une scène 3D naturelle, un site, un musée, pour observer un objet ou une personne sous n'importe quel angle de vue. Il pourrait finalement aussi donner l'idée d'une nouvelle forme de jeux vidéos ou d'expériences interactives faisant usage d'images réelles.

Organisation du papier. Un aperçu de la méthode est donnée en §2. La construction de la variété dans l'espace des paramètres est présentée en §3. La procédure de navigation ou *Sailing* est expliquée en §4. Une série d'expérimentations est proposée en §5. Enfin, en §6 nous dépeignons les multiples perspectives qu'offrent la solution proposée, et donnons une conclusion à ce travail.

2 Scene Sailing

La méthode analyse une vidéo représentant une scène. Lors de la phase d'apprentissage sont identifiées les images voisines et le mouvement les séparant est estimé. Le modèle de mouvement que nous utilisons est l'homographie que l'on considère approximativement correcte dans un voisinage restreint. L'homographie est décrite par 8 paramètres, ce qui conduit par résolution du système des contraintes de mouvement local, à la redistribution des données images dans un espace des paramètre 8-dimensionnel. Pour rendre continue l'exploration de la variété, Les données sont ensuite *kernélisées* à l'aide de fonctions noyau de type gaussien, de telle sorte qu'une variété continue est créée.

L'exploration de la variété est autorisée grâce aux déplacements fournis par l'utilisateur via l'interface capturant le mouvement d'une cible plane qu'il déplace. Pour chaque position visitée dans la variété est calculée l'image qui lui correspond. Pour ce faire on considère un certain nombre de données images proches dans l'espace des paramètres. La position relative à ces données permet de calculer l'homographie à appliquer à chacune de ces données afin qu'elle exprime avec bonne approximation l'image correspondante à la position courante. Les données ainsi transformées par homographie sont fusionnées pour fournir l'image qui est finalement rendue sur l'écran de l'utilisateur. Le processus de navigation est assuré en itérant les opérations d'acquisition du mouvement, d'application du mouvement et mise à jour de la position courante dans l'espace des paramètres, et de la formation de l'image correspondante à cette nouvelle position.

3 Apprentissage

3.1 Topologie de la Vidéo et Capture du Mouvement Local Temporel & Spatial

Nous analysons le mouvement inter-image entre images de la séquence assez voisines pour que le flux optique guidé par homographie soit consistant. Deux images sont considérées voisines ou connectées lorsqu'elles se ressemblent assez. Chaque image est comparée à toutes les autres de la même séquence via la corrélation centrée normalisée. Tout score de corrélation supérieur à 0.85 valide la connection de la paire d'images. La figure 3 représente la matrice des connections qui recense toutes les paires connectées. Pour évaluer le mouvement entre deux images connectées nous estimons l'homographie qui les sépare. L'homographie entre deux images i et j donne lieu à un déplacement 2D des quatre coins de l'image cible. Ces valeurs de déplacement sont stockées dans un vecteur à 8 dimensions v_{ij} qui représente le mouvement homographique. Ce mouvement n'est qu'une estimation du mouvement réel étant donné qu'il ne prend pas en compte le modèle effectif de la

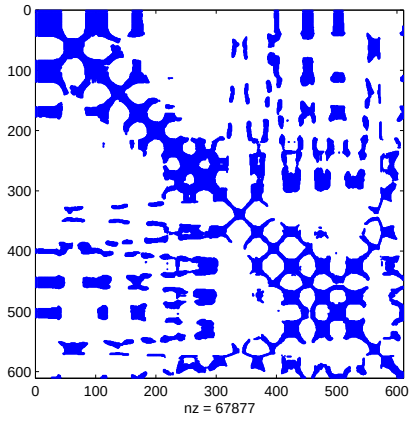

FIG. 3 – Matrice d'adjacences indiquant les connections entre les données. Une connection (en bleu) relie les indices des deux images connectées. Les connections temporelles se trouvent autour et sur la diagonale, et les connections spatiales s'observent hors diagonale : ils indiquent les recoupements de vues qui ont lieu au long de la séquence. Remarquons que $M^T M$ est bien représentée par cette matrice. D'une forme assurant la stabilité à l'inversion, la pseudo-inverse $M^\dagger$ est donc consistante.

scène et ne peut donc calculer le véritable déplacement de caméra.

Nous empilons tous les vecteurs déplacement pour former la matrice V de dimension $m \times 8$, où m est le nombre de paires connectées.

3.2 L'Espace des Paramètres et la Réduction de Dimension

L'idée consiste à calculer un jeu de données s'inscrivant dans un espace 8D que l'on nomme *espace des paramètres* qui respecte au mieux les contraintes réunies dans les vecteurs empilés dans V. Pour cela nous exprimons les déplacements V comme une multiplication entre une matrice de connection extrêmement creuse M de dimension $m \times N$ avec la matrice U de dimension $N \times 8$ qui représente les coordonnées des données dans l'espace des paramètres. N est le nombre total d'images que l'on considère dans la séquence.

Nous ajoutons une contrainte supplémentaire sur les données en augmentant M d'une ligne de $1s$ et V d'une ligne de $0s$. Ceci force la sommes des données à équivaloir au vecteur nul, centrant ainsi ces données.

Le problème linéaire peut s'écrire de cette façon :

$$MU = V \tag{1}$$

Il est très stable et facile à inverser, offrant la solution :

$$U = M^\dagger V \tag{2}$$

avec $M^\dagger$ la pseudo inverse de M.

Les données U ainsi caculées et positionnées dans l'espace des paramètres représentent les images de la vidéo réorganisées en 8D pour exprimer le mouvement mutuel des images voisines. La figure 2 illustre ceci par une projection des données dans l'espace principal de dimension 2.

Pour que l'exploration puisse être effectuée sur un domaine continu, nous appliquons une fonction noyau sur les données pour générer une variété continue qui les enveloppe. Les détails de cette procédure sont fournis plus bas.

4 Navigation

La navigation consiste à permettre à l'utilisateur d'explorer la variété. Dans ce qui suit nous montrons comment l'utilisateur peut se déplacer à l'intérieur de cette variété, guidant ses déplacements au moyen d'une interface capturant le mouvement. Nous montrons comment garder l'exploration à l'intérieur de la variété à chaque instant. Finalement nous expliquons comment construire et afficher l'image qui correspond à chaque position visitée dans la variété.

4.1 Interfaçage de la Navigation

Pour n'importe quelle position courante située à l'intérieur de la variété nous affichons une image interpolée entre les données les plus proches, que l'on déclare *actives*. Nous expliquerons plus en détail cette procédure dans cette section.

Depuis une position initiale, un utilisateur peut fournir une commande de déplacement, c'est à dire un vecteur de déplacement à 8 dimensions, pour permettre au système de calculer et d'afficher la nouvelle vue en prenant en compte les changements requis.

La commande est passée par l'utilisateur au système par le biais d'une interface qui fait l'acquisition en temps réel d'un vecteur de déplacement 8D, permettant ainsi une navigation simple et intuitive à l'intérieur de la variété. L'interface se compose simplement d'une webcam, devant laquelle l'utilisateur déplace un morceau de carton sur lequel est imprimé un motif reconnaissable. La figure 4 illustre le système.

La webcam capture la position des quatre coins 2D d'un motif carré reprojeté sur l'image. Les coordonnées reprojetées courantes sont différentiées avec des coordonnées fixes afin de créer un vecteur déplacement que l'on applique à la position visitée courante afin de la mettre à jour.

De cette façon, la totalité de la variété peut être explorée grâce à ces commandes de déplacement, et au rendu temps réel de l'image correspondante à la position courante visitée. Nous fournissons ainsi le moyen intuitif et immersif d'explorer librement la scène observée, représentée par la variété d'images.

Aux endroits où la variété est dense, l'utilisateur peut naviguer dans toutes les directions de l'espace. Lorsque peu d'images sont connectées, il pourrait n'y avoir qu'une seule direction à explorer.

4.2 Déplacement et Interpolation d'Images

Pour chaque nouvelle position de la variété que l'utilisateur désire visiter, le système doit analyser la position requise, éventuellement la reprojeter sur la variété si celle-ci s'en est écarté, transformer et fusionner les images actives pour finalement rendre l'image finale. Dans la suite nous décrivons les opérations techniques impliquées dans cette procédure globale.

La fonction noyau L'expression du noyau (ou *kernel*) est la suivante :

$$\mathcal{K}(\|v\|) = \begin{cases} \exp\left(\frac{-2\sigma^2\|v\|^2}{(\sigma^2-\|v\|^2)^2}\right) & \text{si } \|v\| < \sigma \\ 0 & \text{sinon} \end{cases}, \quad (3)$$

avec v le vecteur différence entre la position courante et une donnée, et σ le rayon d'enveloppe de la variété, fixé à une valeur suffisament large pour assurer la continuité entre les données connectées, mais limité tout de même pour ne pas violer l'hypothèse de localité pour laquelle le modèle est considéré valide. De cette façon l'image visualisée n'appaîtra pas floue ou bien mal fusionnée. La figure 5 illustre en 2D le manifold construit autour des données.

Pour une position requise u de l'espace des paramètres, on considère la donnée la plus proche (ainsi nous sommes sûrs qu'au moins une donnée est sélectionnée), ainsi que toutes celles placées au plus à une distance équivalente à une proportion de σ.

Appelons *actives* les données sélectionnées ; elles forment l'ensemble $\mathcal{A}$. Pour chaque i^e donnée u_i de $\mathcal{A}$, on calcule le poids donné par le noyau dans l'Equation 3 :

$$w(v_i) = \mathcal{K}(\|v_i\|), \quad (4)$$

où $v_i = u_i - u$.

Les poids définiront la proportion d'images actives à mettre dans le mélange (ou *blend*) final.

Nous devons prêter attention au cas où la position u se trouve à l'extérieur de la variété. On détecte cette situation lorsque la position se trouve à une distance plus grande que

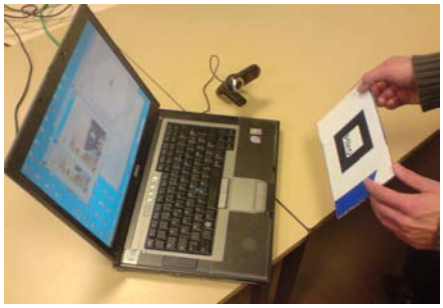

FIG. 4 – Illustration de l'interface utilisateur-navigateur qui se compose d'une webcam effectuant le suivi du mouvement d'un motif rigide et plan. Sur son écran, la machine exécute le rendu de la scène depuis le point de vue courant.

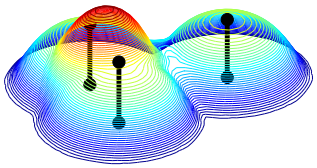

FIG. 5 – Illustration 2.5D de la variété continue construite autour d'un jeu de trois données 2D. Le potentiel cumulé $\sum_{i\in[1\cdot3]} \mathcal{K}(\|v-u_i\|)$ fourni par les noyaux autour des données est représenté dans la troisième dimension par des courbes de niveau en tout point du plan. La variété est définie de façon continue, partout où le potentiel est non nul.

σ de la donnée la plus proche. Dans ce qui suit nous expliquons le type d'action auquel nous avons recours dans ce cas précis.

Conserver le déplacement à l'intérieur de la variété Une nouvelle vue ne peut être construite que si sa position correspondante se trouve à l'intérieur de la variété. Malheureusement les requêtes de déplacement de l'utilisateur portent souvent la nouvelle position u à sortir de la variété. Lorsque cette situation est détectée ($\|u-u_i\| > \sigma$, pour u_i la donnée la plus proche de u), nous reprojetons la donnée contre l'enveloppe interne de la variété.

La reprojection est simplement obtenue par optimisation de la position $\hat{u}$, que l'on cherche la plus proche de u et tel que $\hat{u}$ se trouve sur l'isocourbe de valeur ϵ du noyau. Le problème s'écrit de la façon suivante :

$$\hat{u} = \arg\min \|\hat{u}-u\|, t.q. \sum_{i\in\mathcal{A}} \mathcal{K}(\|\hat{u}-u_i\|) = \epsilon, \quad (5)$$

avec ϵ choisi petit de façon à ce que l'isocourbe considérée soit proche de la surface de la variété. La position est ensuite mise à jour : $u \leftarrow \hat{u}$.

La procédure d'optimisation sous contrainte est entièrement détaillée en annexe, et la figure 6 illustre la reprojection sur la variété.

Warping, blending et affichage A ce stade, nous disposons de toute l'information, les poids et les positions relatives v_i entre u et les u_i, pour construire et afficher l'image finale. Nous commençons par appliquer un *warp* aux images actives. A chaque image est appliquée une compensation de mouvement équivalente à $-v_i$, rapprochant de zéro le mouvement entre toutes les images transformées. Cette procédure d'interpolation mène à l'estimation de l'image (construite à partir des données disponibles utiles) qui correspond à la position u dans l'espace des paramètres. Finalement on fusionne ces images avec une proportion de chaque donnée par son poids normalisé et on affiche finalement l'image résultante.

Le problème de la transformation et de la fusion peut être

écrit de la manière suivante :

$$\mathcal{I}_{blend} = \sum_{i \in \mathcal{A}} \frac{w(v_i)}{Sw} \mathcal{W}(\mathcal{I}_i, -v_i), \quad (6)$$

où $\mathcal{W}(\mathcal{I}_i, -v_i)$ est le warp de l'image $\mathcal{I}_i$ paramétrisé par le vecteur de déplacement $-v_i$. Notons que $Sw = \sum_{k \in \mathcal{A}} w(v_k)$ et normalise les poids attribués à chaque donnée. L'image $\mathcal{I}_i$ correspond à la donnée u_i dans l'espace des paramètres.

5 Expérimentations

Afin d'illustrer le fonctionnement de la méthode décrite en précédence, nous avons fait l'acquisition d'une scène complexe, par des mouvements simples. La scène est filmée en guidant l'appareil numérique par le plan table sur lequel il se trouve. Des translations pures sont effectuées dans le plan, sans appliquer de rotation à l'appareil. Les recoupements entre vue sont correctement assurés et l'espace des paramètres est suffisament dense dans les (deux) dimensions parcourues par la caméra. La séquence se compose d'environ 600 images dont nous détectons les connections mutuelles par corrélation entre chaque paire d'images. Les paires fortement corrélées sont déclarées voisines. Elles sont représentées par la matrice d'ajacences de la figure 3. Le recalage entre les images voisines est réalisé grâce au *Direct Image Registration Toolkit* [8], qui retourne une quantité de mouvement homographique entre paire d'images sous forme d'un vecteur à 8 dimensions. Comme décrit en §3, le mouvement entre paires connectées nous permet d'inférer une distribution des données dans l'espace des paramètres. Ceci définit une variété dans laquelle nous pouvons naviguer. La capture de mouvement par webcam prévue par l'interface que nous utilisons est réalisée grâce à ARToolkit [10] qui retourne à chaque instant la position reprojetée dans l'image des quatre coins d'un motif carré plan.

Une illustration de navigation est illustrée figure 7 : un extrait d'images provenant de l'expérience de *Sailing* est proposé. Lors du *Sailing* nous superposons quatre flèches à l'image courante en guise de retour à l'utilisateur de la commande homographique qu'il est en train de fournir. L'équivalent des déplacements au sein de l'espace des paramètres est illustré sur la figure 8.

Dans le contexte du mouvement à dimensions contraintes de la caméra, les résultats obtenus sont probants sur une scène de haute complexité que des méthodes de reconstruction ont du mal à traiter.

Nous cherchions initialement à naviguer dans une scène acquise manuellement, et filmée de façon aléatoire depuis de nombreux points de vue. En pratique il s'avère difficile d'obtenir des résultats concluants par cette méthode dû au fait que les images ne remplissent pas assez bien l'espace des paramètres de façon localement dense. Le remplissage dense des 8 dimensions de cet espace nécéssite de mettre en place une stratégie plus élaborée que celle de simplement filmer manuellement la scène. L'acquisition méthodique par bras mécanique peut être une solution, mais son utilisation fournirait facilement les paramètres réels de poses des caméras, ce qui rendrait obsolète notre approche. Nous exposons une autre possibilité dans les perspectives.

6 Perspectives et conclusion

Dans ce papier nous introduisons le *Scene Sailing*. Cette méthode permet de naviguer intuitivement au sein d'une collection d'images représentant une même scène. La méthode est capable de traiter des scènes complexes qui mettent en défaut les algorithmes de reconstruction. Par ce premier travail, nous montrons empiriquement que la connaissance des caméras n'est pas nécessaire pour la navigation. Une connaissance plus restreinte, le mouvement entre les vues proches, suffit à déterminer correctement une répartition des images dans un espace des paramètres dans lequel il est très simple de naviguer. La bonne paramétrisation des données dans cet espace requière que la vidéo soit dense dans les dimensions de mouvements à explorer, et que les vues se recoupent à des moments variés de la séquence, afin de "fermer les boucles" et de bien contraindre la distribution dans ces dimensions.

Une expérience vient valider le principe dans le contexte de mouvements d'acquisition vidéo contraints dans principalement deux directions.

L'objectif final serait celui d'implanter un système qui procède en temps réel non seulement au *Sailing*, mais également à l'apprentissage. Le système fournirait un retour immédiat à l'utilisateur filmant la scène en lui indiquant quelles sont les régions de l'espace méritant d'être mieux définies pour assurer que la distribution des données soit correcte dans l'espace des paramètres. L'utilisateur n'aurait alors qu'à compléter la saisie en des points de vue bien spécifiés par le système, qui lui indiquerait en temps réel par indications visuelle et sonore sa distance à la zone à renseigner. Par exemple, un son continu serait émis lorsque les images acquises par la caméra apportent de l'information, alors qu'un son discontinu à fréquence variable indiquerait le rapprochement ou l'éloignement de la zone informative selon que cette fréquence augmente ou diminue. Le système devra aussi pouvoir ne sélectionner que les images nécessaires à la bonne définition de la variété, et éviter trop de redondance.

Annexe

Le problème d'optimisation sous contrainte posé en équation 5 peut se réécrire sous la forme du Lagrangien :

$$\mathcal{L}(\hat{u}, \lambda) = \|\hat{u} - u\|^2 + \lambda \left[\sum_{u_i \in \mathcal{A}} \mathcal{K}(\|\hat{u} - u_i\|) - \epsilon \right] \quad (7)$$

On écrira $\mathcal{K}_i(\hat{u}) = \mathcal{K}(\|\hat{u} - u_i\|)$.

La règle de mise à jour puis l'expansion de Taylor prévues

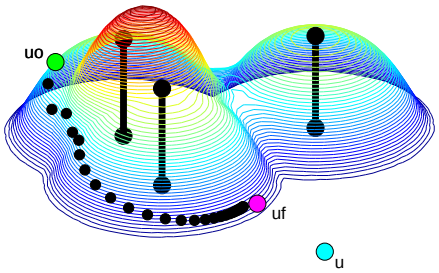

FIG. 6 – Illustration de la procédure d'optimisation contrainte. Le cas des trois données 2D est présenté de nouveau. On désire projeter la position u sur la variété. Le point d'initialisation $\hat{u}_0$ se déplace itérativement jusqu'à atteindre $\hat{u}_f$, position de l'isocourbe de valeur ϵ la plus proche de la position u. Si $\hat{u}_0$ est loin de $\hat{u}_f$ pour cette illustration, en pratique on place $\hat{u}_0$ au plus proche de la solution en cherchant le point d'intersection de l'isocourbe de valeur ϵ et du segment $[u, u_1]$, où u_1 est la donnée la plus proche de u.

par Gauss-Newton nous donne la forme suivante :

$$\begin{aligned}\mathcal{L}(\hat{u}+\delta,\lambda) &\approx \|\hat{u}+\delta-u\|^2 \\ &\quad +\lambda\left[\textstyle\sum_{u_i\in\mathcal{A}}(\mathcal{K}_i(\hat{u})+\mathcal{K}_i^{'}(\hat{u})^T\delta)-\epsilon\right],\end{aligned} \tag{8}$$

avec $\mathcal{K}_i^{'}$ dérivée de $\mathcal{K}_i$ par rapport à $\hat{u}$. Ecrivons que $\widetilde{\mathcal{L}}(\hat{u}+\delta,\lambda)$ est égal strictement au terme de droite.
Les dérivées de $\widetilde{\mathcal{L}}$ s'annulent à chaque itération. On a

$$\frac{\partial\widetilde{\mathcal{L}}}{\partial\delta}(\hat{u}+\delta,\lambda)=2\delta+2(\hat{u}-u)+\lambda\sum_{u_i\in\mathcal{A}}\mathcal{K}_i^{'}(\hat{u}), \tag{9}$$

$$\frac{\partial\widetilde{\mathcal{L}}}{\partial\lambda}(\hat{u}+\delta,\lambda)=\sum_{u_i\in\mathcal{A}}\left(\mathcal{K}_i(\hat{u})+\mathcal{K}_i^{'}(\hat{u})^T\delta\right)-\epsilon, \tag{10}$$

toutes deux égalant 0.
Ce qui conduit au système suivant :

$$\begin{pmatrix} I & k \\ k^T & 0 \end{pmatrix}\begin{pmatrix} \delta \\ \lambda \end{pmatrix}=\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \tag{11}$$

avec $k=\frac{1}{2}\sum_{u_i\in\mathcal{A}}\mathcal{K}_i^{'}(\hat{u})$,
et $b_1=u-\hat{u}, \quad b_2=\frac{1}{2}(\sum_{u_i\in\mathcal{A}}\mathcal{K}_i(\hat{u})-\epsilon)$.
L'inverse du bloc $\begin{pmatrix} I & k \\ k^T & 0 \end{pmatrix}$ est de forme $\begin{pmatrix} A & d \\ d^T & c \end{pmatrix}$.
Multipliées l'une à l'autre ces deux matrices donnent l'identité. L'inverse peut donc être déterminée par résolution du système suivant :

$$\begin{aligned} A+kb^T &= I \\ a^Tk &= 1 \\ a^TA &= 0^T \\ k+ac &= 1 \end{aligned} \tag{12}$$

Finalement l'inverse vient simplement :

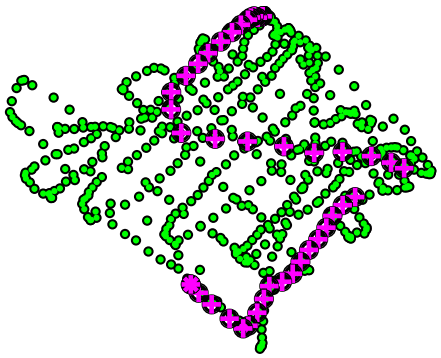

FIG. 8 – Trajet dans la variété correspondant à la succession d'images de la figure 7. Le départ se trouve au point de parcours le plus haut. L'étoile indique la dernière vue.

$$\begin{pmatrix} I-\frac{kk^T}{||k||^2} & \frac{k}{||k||^2} \\ \frac{k^T}{||k||^2} & -\frac{1}{||k||^2} \end{pmatrix} \tag{13}$$

On remarquera que le calcul de δ ne dépend pas de λ et qu'on a donc pas besoin de calculer ce dernier (cette intéressante propriété advient lorsque la contrainte est linéaire en δ).
La solution à ce problème d'optimisation sous contrainte revient donc simplement à itérer sur ces deux opérations :

$$\begin{aligned} \delta &= (I-\tfrac{kk^T}{||k||^2})b_1+\tfrac{k}{||k||^2}b_2, \\ \hat{u} &\leftarrow \hat{u}+\delta. \end{aligned} \tag{14}$$

Références

[1] E. Adelson, J. Bergen. *The plenoptic function and the elements of early vision.* Cambridge, Massachusetts : MIT Press.

[2] N. Snavely, S.M. Seitz, R. Szeliski. *Photo tourism : exploring photo collections in 3D.* ACM Trans. Graph., 2006.

[3] S. Chen, L. Willliams. *View interpolation for image synthesis.* SIGGRAPH, 1993.

[4] P. Dollar, V. Rabaud, S. Belongie. *Learning to Traverse Image Manifolds.* UCSD Technical Report, 2007.

[5] S. J. Gortler, R. Grzeszczuk, R. Szeliski, M. F. Cohen. *The Lumigraph.* SIGGRAPH, 2006.

[6] M. Levoy, P. Hanrahan. *Light Field Rendering.* SIGGRAPH, 1996.

[7] A. Fusiello. *Specifying virtual cameras in uncalibrated view synthesis.* IEEE Trans. on Circuits and Systems for Video Technology, 2007.

[8] A. Bartoli. *Groupwise Geometric and Photometric Direct Image Registration.* PAMI, 2008.

[9] M. Lhuillier, L. Quan. *Image Interpolation by Joint View Triangulation.* CVPR, 1999.

[10] *ARToolkit.* http ://www.hitl.washington.edu/artoolkit/

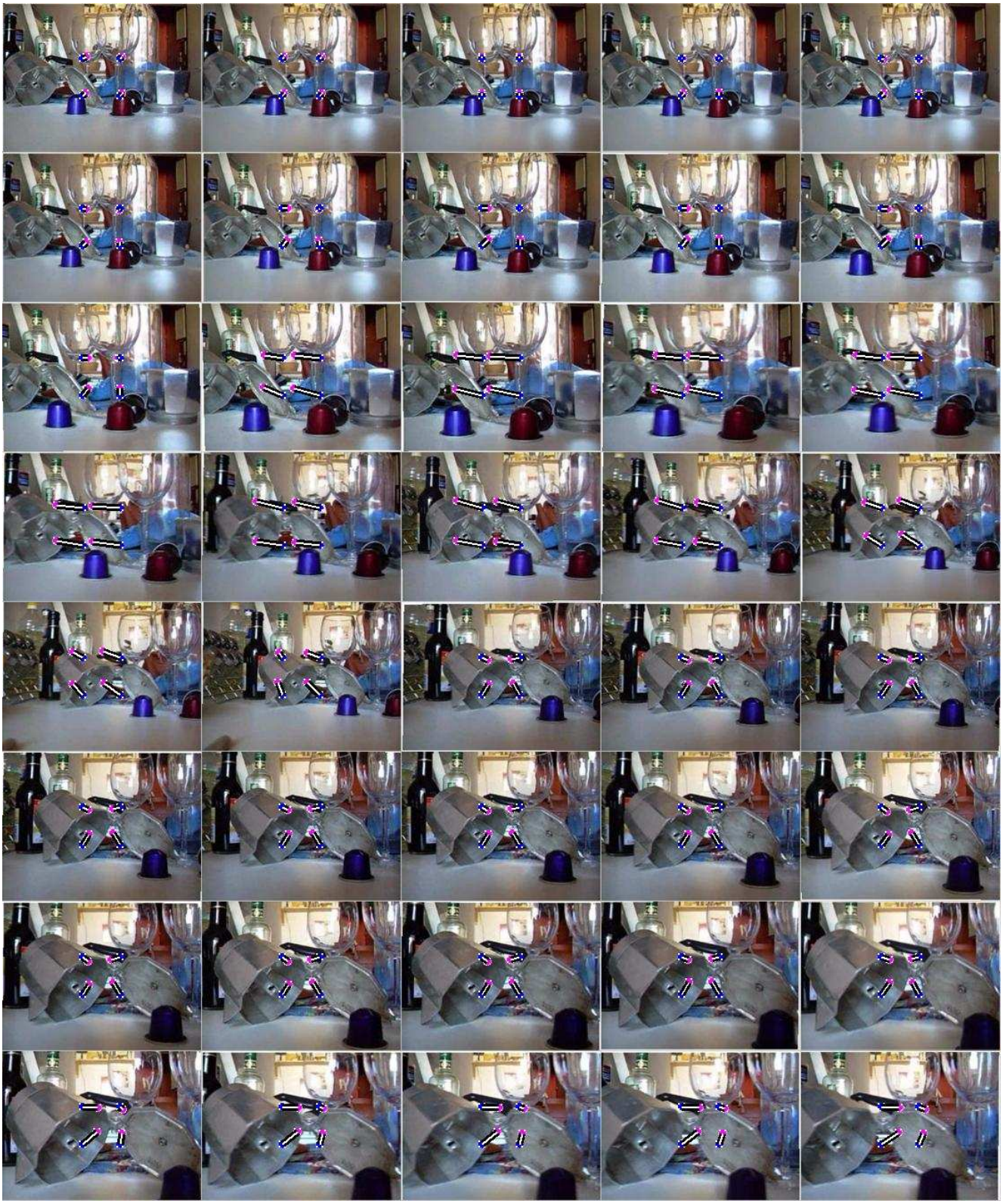

FIG. 7 – Extrait d'une séance de *Sailing*. Les commandes de l'utilisateur sont représentées par quatre flêches dont les origines sont fixes, et les extrémités suivent le mouvement des coins du motif plan manoeuvré par l'utilisateur devant la webcam.