Suivi tridimensionel de la main

Hand's Posture And Gesture

F. Le Jeune[1] R. Deriche[2] R. Keriven[1] P. Fua[3]

[1] CERMICS, ENPC, Marne La Vallée, FRANCE
[2] INRIA Sophia Antipolis, FRANCE
[3] CVlab, EPFL, SUISSE

lejeune@cermics.enpc.fr Rachid.Deriche@sophia.inria.fr
keriven@cermics.enpc.fr pascal.fua@epfl.ch

Résumé

La position et la configuration d'une main en mouvement ne sont pas faciles à retrouver pour un ordinateur équipé de caméras. Cela est en partie dû au nombre élevé de degrés de liberté de cet objet ainsi que de sa forme complexe. Nous avons basé notre méthode de suivi sur des nuages de points en trois dimensions obtenus par stéréovision afin de conserver l'information de profondeur. L'aspect complexe et organique de la main a été modélisé par un squelette articulé habillé de metaballs. Quant à la difficulté liée au grand nombre de degrés de liberté, nous nous sommes efforcés de la réduire en rajoutant des contraintes articulatoires. La recherche de la position correcte à partir de la position précédente estimée repose sur la minimisation d'une énergie basée sur la distance entre le modèle de la main et le nuage de points en trois dimensions.

Mots Clef

Vision par ordinateur, suivi de la main, modèle articulé, contraintes

Abstract

The hand with its pose and gesture is not easy to track with cameras. It is mostly because of a high number of degrees of freedom and a complex shape. Our algorithm is based on three dimensional clouds of points issued from stereovision in order to keep the depth information. The complex and organic aspect of the hand is modeled by an articulated skeleton on which metaballs are attached. We reduce the difficulty of the problem by adding joint constraints to the model. The core of our program is an energy-based minimization which fits the model at each frame using the distance between the model and the clouds of points, the result at the previous frame is chosen as an initial guess.

Keywords

Computer vision, hand, 3D model, constraints

1 Problématique

Le suivi de la position de la main ainsi que de la configuration des doigts est un domaine en pleine expansion ces dix dernières années. Il emprunte différentes voies : une seule caméra ou bien plusieurs, analyse de la silhouette ou de la texture de la main, en couleur ou en noir et blanc, à partir d'un modèle articulé ou d'une base de données de positions prédéfinies. Chaque méthode a ses avantages et ses inconvénients et il n'y en a pour l'instant aucune qui soit parfaite.

Nous avons fait le choix d'un système basé sur plusieurs caméras afin de pouvoir lever les ambiguïtés de profondeur qui apparaissent avec les systèmes mono-caméra. Nous travaillons à partir de nuages de points tridimensionnels issus de stéréovision que nous essayons de mettre en correspondance avec un modèle articulé tridimensionnel de la main. Nous avons fait le choix de l'utilisation d'un modèle articulé et non d'une méthode utilisant sur des bases de données afin d'être en mesure de pouvoir suivre toutes sortes de positions de la main sans nécessiter d'apprentissage.

2 Acquisition de données

Dans un premier temps, nous obtenons nos nuages de points à partir de caméras pré-calibrées conçues pour la stéréovision : les *Digiclops* de la société *Pointgrey* [11] accompagnés de leur API propriétaire de stéréovision. Un dispositif simple et facile d'utilisation mais peu configurable.

L'utilisation de l'information de profondeur permet de faire un filtrage rapide du nuage de points en ne sélectionnant que les points situés dans un intervalle de distances correspondant à la zone où se trouve la main.

Une autre technique de filtrage [8] est utilisée pour supprimer des points inutiles : un filtre sur la couleur. La couleur (R, G, B) est convertie dans un espace de couleurs à deux

dimensions (r, g) selon :

$$\begin{cases} r = \frac{R}{R+G+B} \\ g = \frac{G}{R+G+B} \end{cases}$$

La zone qui correspond à la couleur de la peau se trouve à l'intérieur d'une ellipse (figure 1). Les paramètres de cette dernière (taille initiale, rapport entre les axes, orientation) ont été calculés à partir d'histogrammes de couleur sur des zones manuellement segmentées dans des conditions d'éclairage similaires (une ou deux images suffisent). La tolérance de ce filtre se règle en modifiant la taille de l'ellipse.

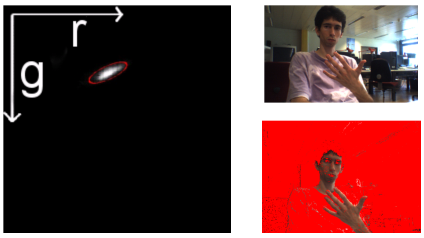


FIG. 1 – zone correspondant à la couleur de la peau dans l'espace (r, g) et application du filtre sur une image.

Ces deux filtres permettent de supprimer une partie des points qui ne correspondent pas à la main mais ne permettent pas de corriger tous les points erronés. En pratique, on obtient avec des *Digiclops* des données incomplètes, bruitées et d'assez faible résolution (320×240). Nous pensons que la principale source d'imperfection de ces données est l'utilisation des caméras en dehors de leur domaine de fonctionnement habituel (leur focale fixe par exemple est réglée pour voir des objets à plusieurs mètres et non à quelques dizaines de centimètres).
En attendant de pouvoir utiliser un système d'acquisition de haute qualité, nous avons utilisé un générateur de nuages de points qui, à partir des paramètres du modèle de la main et de la caméra, génère un nuage de points artificiels correspondant à la surface du modèle. Nous avons cherché une autre source pour obtenir nos nuages de points. Afin de travailler dans des conditions assez proches de celles que nous aurions obtenues avec des caméras réelles, nous sommes passés à l'utilisation de séquences vidéos de synthèses générées par le logiciel *Poser* [12] et représentant une main articulée dans une séquence d'une centaine d'images selon deux points de vues assez proches (séquence fournie par le groupe *MOVI* de l'Inria Grenoble [1]) (figure 2).
A partir de ces images, nous utilisé les mêmes programmes de stéréovision que ceux que nous pensons utiliser lorsque nous aurons nos propres caméras haute qualité. Comme dans [4], nous utilisons un algorithme basé sur la corrélation [9]. Nous réduisons ensuite le nombre de points 3D obtenus pour n'en garder qu'entre 3.000 et 10.000. La sélection des points retenus est pour l'instant arbitraire mais nous pensons n'en conserver qu'un nombre fixé par région 3D lorsque nous utiliserons des images réelles.

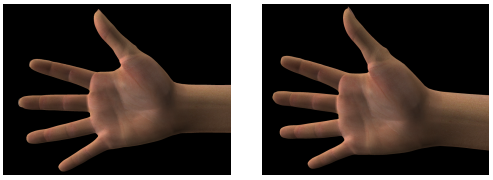


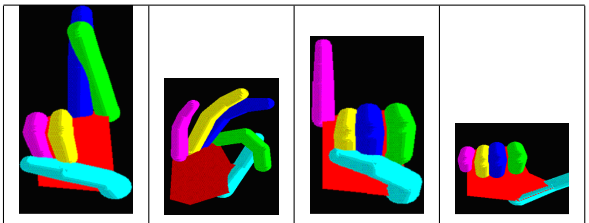
image caméra 1. image caméra 2.

FIG. 2 – images extraites de la séquence de synthèse.

3 Modèle de la main

3.1 Squelette articulé

Pour créer notre modèle articulé de la main, nous nous sommes inspirés à la fois des résultats de Q. Delamarre et O. Faugeras[2] ainsi que de ceux de J. Rehg et T. Kanade[3]. Lorsqu'il ne subit aucune contrainte, ce modèle a 26 degrés de liberté. Le modèle a été validé par une application qui affiche les différentes lettres de l'alphabet de la langue des signes française(figure 3).



"R"	*"F"*	*"I"*	*"A"*

FIG. 3 – exemples de positions de notre modèle articulé

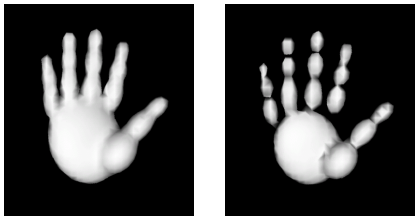
3.2 Ellipsoïdes et *metaballs*

L'habillage de ce squelette articulé est réalisé par des *metaballs* : on place sur le squelette un certain nombre d'ellipsoïdes qui relient différentes articulations entre elles dessinant ainsi les doigts. Nous avons seize ellipsoïdes au total : une pour la paume et trois ellipsoïdes par doigt.
La surface de la main est alors une isosurface de la somme de fonctions de potentiel[6]. Nous avons choisi l'exponentielle des distance aux ellipsoïde comme fonctions de potentiel et l'isosurface de niveau 1.

$$d(X, \alpha) = -\log\left(\sum_{e=1}^{nbEllipses} \exp^{-dist_e(X,\alpha)}\right)$$

où $d(X, \alpha)$ est la distance signée entre le point X et la surface de la main dont la position est paramétrisée par le vecteur α. $\mathrm{dist}_e(X, \alpha)$ est la distance signée entre le point X et la surface de l'ellipse e composant le modèle de la main.
Utiliser des *metaballs* et donc d'estimer la distance au modèle par une fonction assez complexe permet d'obtenir un aspect plus lissé, réaliste et organique que si nous nous étions contentés d'estimer la distance au modèle par la distance à l'ellipsoide le plus proche (comme le montre la figure 4).

Afin d'éviter la "fusion" des doigts entre eux (phénomène classique lors de l'utilisation de *metaballs*), la fonction d n'est en réalité calculée que comme la somme des fonctions distance à certains ellipsoïdes. Le choix de ces ellipsoïdes est fait en fonction du point X : il s'agit de l'ellipsoïde dont la surface est la plus proche de X ainsi que tous les ellipsoïdes qui y sont reliés dans le modèles articulé.



metaballs ellipsoïdes

FIG. 4 – surface du modèle selon l'estimation de la distance choisie

3.3 Distance à un ellipsoïde

Le problème du calcul de la distance d'un point à la surface d'un ellipsoïde n'est pas simple. Cette mesure doit être approchée. La difficulté réside dans le fait que bien que l'intersection entre le segment reliant le point (noté P sur le schéma) au centre de l'ellipsoïde et la surface de l'ellipsoïde soit un point aisément calculable (point I sur le schéma), ce n'est pas à partir de cette intersection que la distance doit être calculée. Le point de la surface de l'ellipsoïde qui minimise la distance au point P est le point D sur le schéma. Les coordonnées de ce point ne sont pas calculables à l'aide de fonctions simples

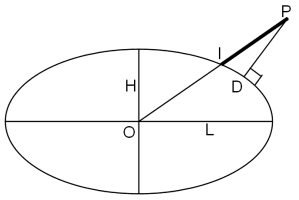


FIG. 5 – approximation de la distance PD par PI.

Une première estimation consiste à appliquer un changement de base à l'ensemble $\{\text{point}P, \text{ellipsoïde}\}$ afin de transformer l'ellipsoïde en une sphère et le point P en un point P'. La distance du point P' à la surface de la sphère est alors facilement calculable et est utilisée comme approximation de la distance du point P à la surface de l'ellipsoïde. Cette distance est notée $d_1(x, y)$ sur la figure 6.
On remarque que cette valeur décroît plus vite selon le petit axe de l'ellipsoïde que selon son plus grand axe. Afin de rendre cette décroissance constante au moins selon les trois axes de l'ellipsoïde, nous utilisons une approximation différente en calculant la distance PI, comme suggéré dans [1] (figure 5). Raisonnons en 2 dimensions pour simplifier les calculs et supposons l'ellipse centrée à l'origine.

Soit $M = \begin{bmatrix} \frac{1}{L} & 0 \\ 0 & \frac{1}{H} \end{bmatrix}$ la matrice qui transforme l'ellipse (L, H) en cercle unitaire. Le point I est l'intersection de la droite (OP) et de l'ellipse. Soient I' et P' les images respectives de I et P par cette transformation. L'image de O restant O, on obtient donc que I' est l'intersection de (OP') et du cercle unitaire.

$$I' = \frac{1}{|P'|} \begin{pmatrix} \frac{x}{L} \\ \frac{y}{H} \end{pmatrix}$$

Ceci nous permet de calculer les coordonnées du point I.

$$I = M^{-1}.I' = \frac{1}{|P'|} \begin{pmatrix} x \\ y \end{pmatrix}$$

La distance signée IP s'obtient alors aisément et au final :

$$d_2(x, y) = \sqrt{x^2 + y^2}. \left(1 - \frac{1}{\sqrt{\left(\frac{x}{L}\right)^2 + \left(\frac{y}{H}\right)^2}} \right)$$

Même si cette valeur est encore une approximation, elle a le mérite d'être un compromis acceptable entre une bonne estimation de la distance réelle à l'ellipsoïde et une distance facilement calculable. Les dérivées de cette distance existent à proximité de la surface et sont facilement calculables. Nous les utilisons dans l'algorithme de minimisation.
Ceci vaut pour l'extérieur de l'ellipsoïde : à l'intérieur d_2 fait apparaître des lignes d'iso-valeurs assez différentes de celles de la vraie distance. Nous utilisons alors directement la distance au centre de la sphère et lui appliquons une fonction affine telle que l'approximation de la distance vaille, telle la vrai distance signée, 0 à la surface et $-r$ au centre (avec $r =$ dimension du plus petit des trois axes).
On utilise finalement :

$$d_3(x, y) = \begin{cases} \left(\sqrt{\left(\frac{x}{L}\right)^2 + \left(\frac{y}{H}\right)^2} - 1 \right).r & \text{à l'intérieur} \\ d_2(x, y) & \text{à l'extérieur} \end{cases}$$

Bien qu'ayant dû utiliser une approximation de la distance à la surface des ellipsoïdes, le résultat final est satisfaisant (figure 6). De plus, le fait d'avoir approché cette distance par une fonction mathématique presque partout dérivable nous permet d'obtenir de façon analytique, pour chaque point de l'espace, sa distance à la surface de la main ainsi que la dérivée de cette distance par rapport à chacun des paramètres du modèle de la main. comme [4], ce résultat nous permet d'utiliser l'algorithme de minimisation de Levenberg-Marquardt [10] pour trouver les paramètres minimisant la distance du nuage de points issu de nos caméras à la surface du modèle de la main.

3.4 Contraintes

Afin de restreindre cet algorithme à un espace de positions réalistes, notre modèle de main s'est vu adjoindre deux types de contraintes sur les articulations.

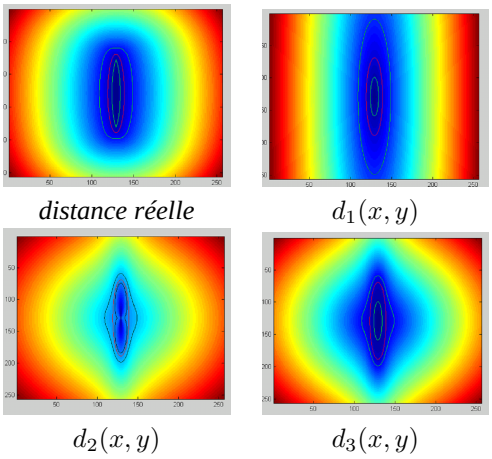
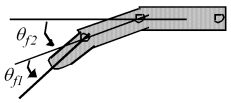


FIG. 6 – distances approchées.

Les premières sont de type *min/max* et donnent des limites pour chaque articulation indépendamment des autres. A chaque pas de temps de l'algorithme de Levenberg-Marquardt, ces contraintes sont testées et chaque paramètre qui sort de sa zone de contrainte est reprojeté sur la borne la plus proche de son intervalle autorisé. Ces contraintes sont simples et permettent d'améliorer nettement la qualité du suivi.
Les autres contraintes sont des contraintes qui permettent d'obtenir la valeur d'un paramètre directement à partir de celle d'un autre. Elles sont essentiellement placées sur la dernière articulation de chaque doigt où la valeur de l'articulation est fixée à deux tiers de la valeur de l'articulation précédente (figure 7). Cette contrainte que l'on retrouve dans [2] est assez souvent utilisée en suivi de la main à partir de modèles articulés et permet de supprimer quatre degrés de liberté sur les 26.



$$\theta_{f_1} = \frac{2}{3}.\theta_{f_2}$$

FIG. 7 – contrainte sur la dernière articulation.

Nous ajoutons aussi une contrainte de ce type sur les valeurs d'abduction des quatre doigts (l'abduction est la possibilité d'écarter les doigts). Elles sont liées ensemble par l'équation :

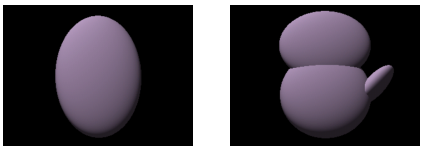
$$\left[\begin{array}{cccc} c_0 & c_1 & c_2 & c_3 \end{array}\right]^t.\alpha = \left[\begin{array}{cccc} A_0 & A_1 & A_2 & A_3 \end{array}\right]$$

où A_0, A_1, A_2, A_3 correspondent respectivement aux valeurs d'abduction de l'index, du majeur, de l'annulaire et de l'auriculaire et c_0, c_1, c_2 et c_3 sont des constantes propres à chaque personne et mesurées avant le suivi. Cette contrainte biologique permet de remplacer 4 valeurs d'abduction par un unique paramètre α. Nous n'avons pas la connaissance de l'utilisation de cette contrainte par d'autres équipes de chercheurs et nous ne pouvons certifier qu'elle est légitime, mais les premiers tests semblent indiquer qu'elle est réaliste et aide au suivi ; elle permet de faire finalement baisser le nombre de degrés de liberté du modèle à 18 (décomposés en 6 degrés pour la position dans l'espace et 12 autres pour le geste).
Des contraintes de type *min/max* dont les bornes *min* et *max* sont des fonctions d'autres paramètres sont a l'étude. Elles permettraient de modéliser le fait que les valeurs possibles de l'angle d'abduction se réduisent lorsque les doigts se replient vers la paume (à partir d'une flexion de 90 degrés, il n'y a plus d'abduction possible).

3.5 Calibration du modèle

Elle est pour l'instant faite de façon manuelle à partir des séquences vidéo. Nous envisageons différentes manières de rendre cette tâche un peu plus automatisée.
Lorsque la qualité des données ne nous permet pas de suivre correctement le modèle complet de la main, nous avons envisagé la possibilité d'utiliser 2 autres modèles (figure 8) :

1. Le plus simple n'est composé que d'une ellipse. Il comporte six degrés de liberté et devrait nous permettre de retrouver ces six paramètres lorsque que le nuage de points n'est pas suffisamment complet. En pratique, il apparaît que ce sont surtout les coordonnées du centre de l'ellipse qui sont retrouvées, son orientation ne correspondant pas vraiment toujours à l'orientation de la main. Cette liberté d'orientation est surtout dûe à la faible ressemblance entre ce modèle composé d'une ellipse unique et une main réelle.
2. Nous résolvons alors le problème en utilisant un modèle intermédiaire (présenté dans l'image ci-dessous dans une version n'utilisant pas les *metaballs*). Composé de trois ellipses et ayant la forme d'une moufle, il reste suffisamment simple pour avoir un nombre réduit de degrés de liberté (9 degrés de liberté) et pour avoir des ellipsoïdes suffisamment grands pour ne pas trop se faire influencer par des absences partielles de données dans le nuage de points (essentiellement des doigts que le programme de stéréovision utilisé n'arriverait pas à reconstruire). Les tests sont concluants et permettent un suivi approximatif de la main quand le modèle totalement articulé ne fonctionne pas.



6 degrés de liberté 9 degrés de liberté

FIG. 8 – modèles simplifiés (ici sans *metaballs*)

4 La recherche de la solution

Le suivi de la main se fait image par image, la position initiale pour l'image n étant la position estimée pour l'image

$n-1$. Pour l'instant la position initiale de la séquence est fixée manuellement.
L'algorithme utilisé pour la minimisation est celui de Levenberg-Marquardt [10]. Il sert à minimiser des fonctions pouvant s'écrire sous la forme :

$$F(\alpha) = \sum_{i=1}^{n} \left(\frac{f(i,\alpha) - y_i}{\sigma_i} \right)^2$$

Ici $f(i,\alpha) = d_3(X_i,\alpha)$ où i parcourt les n points du nuage, $y_i = 0$, $\sigma_i = 1$ et α est le vecteur contenant les paramètres du modèle. L'utilisation de cette méthode de suivi est le résultat d'une collaboration entre l'INRIA, l'ENPC et l'EPFL et prolonge les travaux décrits dans [7].
Une des raisons pour lesquelles la solution fournie par l'algorithme ne correspond pas à la position réelle de la main est qu'il a été piégé dans un minimum local. Une situation de minimum local que l'on rencontre fréquemment est lorsque chacun des 4 doigts va se coller sur le nuage de points correspondant à son voisin. Une solution pour sortir de ce genre de minimum local consiste à relancer une nouvelle séquence d'optimisation à partir de deux autres positions consistant à effectuer une rotation décalant les doigts (voir figure 9).

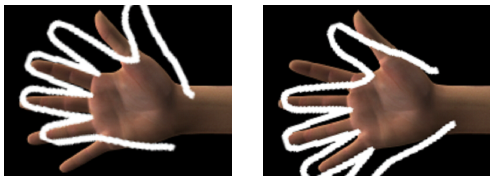


FIG. 9 – minima locaux.

Cette méthode a prouvé qu'elle fonctionnait mais peut aussi involontairement diriger le modèle vers une mauvaise position si le nuage de points est assez imparfait. En pratique, elle résout plus de problèmes qu'elle n'en crée.

5 Visualisation des résultats

Pour visualiser notre surface définie implicitement, nous discrétisons en une grille régulière la zone où se trouve la main. Nous calculons alors la valeur de la fonction distance en chacun des noeuds de la grille puis nous utilisons l'algorithme des *marching cubes* [5] et affichons le résultat.
Le principal inconvénient de cette méthode est la durée du calcul de la fonction distance en chacun des noeuds. Elle ne peut donc pas être utilisée pour un affichage du modèle dans un programme permettant de modifier interactivement ses paramètres. Une solution dégradée consiste alors à dessiner directement les ellipsoïdes qui ont servi de base aux *metaballs*. Nous utilisons cet affichage dans le module de placement initial du modèle ainsi que pour l'affichage de l'animation du suivi.
L'affichage basé sur l'algorithme des *marching cubes* est lui utilisé pour visualiser avec précision le résultat du suivi.

6 Résultats expérimentaux

Notre algorithme a été testé sur deux types de séquences.
Les premières sont des séquences réelles prises à l'aide des *Digiclops* (résolution 320×240) qui s'avèrent ne pas être de bonne qualité car les *Digiclops* semblent avoir été utilisés dans des conditions inadéquates. Le suivi du modèle complet de la main est un échec. La cause première est le nuage de points qui est très loin de correspondre à la réalité (des zones de la main n'ont pas de points 3D, d'autres ont des points dont la profondeur est fausse de près d'un centimètre). Le suivi d'un modèle de la main simplifié en forme de moufle (9 degrés de liberté et un ensemble de quatre doigts représentés par un unique ellipsoïde fig. 8) fonctionne beaucoup mieux.
L'autre type de séquence est une séquence en images de synthèse de résolution 768×576 (comme écrit plus haut, elle nous a été fournie par le groupe *MOVI* de l'Inria Grenoble [1]). Le suivi à l'aide d'un modèle complet totalement articulé se passe assez bien tant qu'il n'y a pas de trop grosses occlusions (figure 10).
Notre méthode fonctionne donc en mode dégradé sur des images réelles de faible qualité et convenablement sur des images de synthèse. Nous sommes en train de procéder à des tests sur des images réelles de qualité standard sur lesquelles les algorithmes usuels de stéreovision fonctionne habituellement.
Le temps mis par l'algorithme pour un positionnement du modèle complet sur un nuage dense (10.000 points en moyenne) et 100 itérations de Levenberg-Marquardt par image est de l'ordre de 10s par image sur un Intel Xeon 2.4GHz. Ce temps n'est pas extrêmement long et pourrait être facilement rabaissé en utilisant moins de points (environ 50% moins) et un nombre moindre d'itérations (autour de 20 itérations au lieu de 100). Mais il ne semble pas pouvoir donner des performances temps-réel dans un avenir proche. Nous nous attendons plutôt à une vitesse de l'ordre d'une demi-seconde par image.

7 Conclusion

Nous avons présenté dans cet article une méthode de suivi de la main basée sur un modèle articulé et sur un nuage de points en trois dimensions calculé à partir de plusieurs caméras. Afin de rendre plus simple ce problème complexe, nous y avons ajouté des filtres sur le nuage de points (couleur, distance à la caméra) et des contraintes sur les paramètres du modèle (valeurs minimales et maximales, dépendances linéaires entre certains paramètres). Les améliorations des résultats grâce à l'ajout de ces contraintes sont nettement visibles. L'utilisation d'une approximation réaliste de la distance entre un point et un ellipsoïde montre aussi des résultats bien meilleurs qu'avec les fonctions plus simples que nous utilisions par le passé.
Cependant les résultats ne sont pas parfaits, surtout lorsque le nuage de points n'est plus assez dense autour de certains doigts. Afin d'améliorer cela, nous envisageons diverses possibilités.

Tout d'abord, la gestion de la continuité des paramètres d'une image à une autre (par filtrage ou lissage) éviterait de voir le modèle "sauter" sur certaines images mal suivies. Cela permettrait aussi d'obtenir des estimations de mouvement afin de continuer le suivi de certaines parties de la main qui se retrouvent temporairement occultées. Ceci dit, cette méthode n'est pas à l'abri de demi-tours éventuels dans le mouvement de la main ou des doigts.
Nous pensons aussi inclure l'avant-bras dans le modèle et dans la procédure d'acquisition afin de fixer le poignet et éviter certains glissements ou certaines rotations de la main.
Nous pourrions aussi rajouter quelques contraintes articulaires sur le modèle de la main.
Et enfin, nous envisageons d'utiliser un peu plus les images 2D des caméras qui nous servent actuellement uniquement pour la reconstruction du nuage de points et pour le filtrage des points de couleur "peau", en rajoutant dans le processus de minimisation l'information sur la distance entre les contours de l'image et les contours projetés du modèle (comme nous l'avions fait en [11]).
Il reste aussi le problème de la position initiale. Une solution envisageable serait de forcer l'utilisateur à avoir en début de séquence une position facilement identifiable comme la main ouverte, paume face à la caméra.

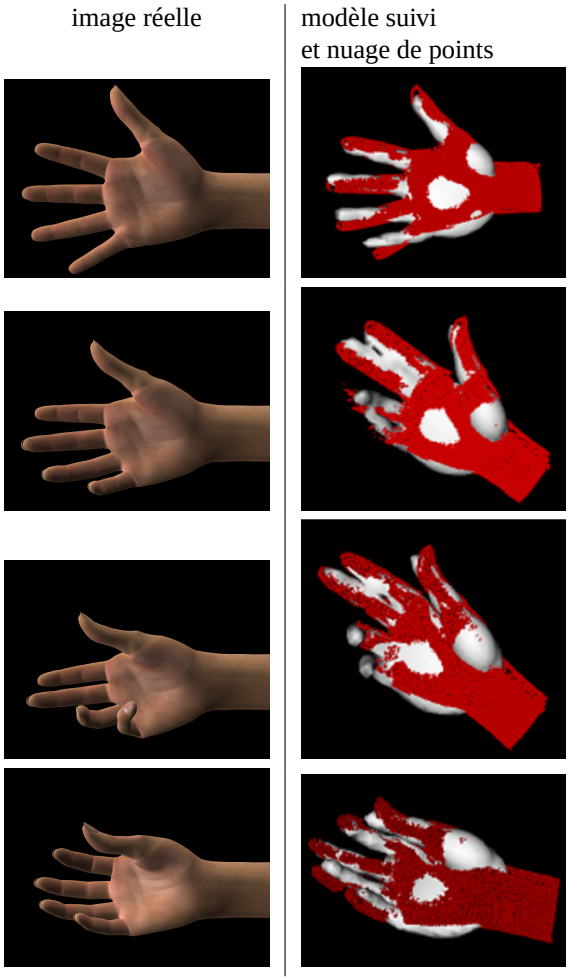


FIG. 10 – Suivi sur une séquence avec un modèle de la main totalement articulé (images à $t = 0, 20, 40, 60$ et 80). A gauche, les images en entrée, à droite, le nuage de points en rouge et le modèle articulé retrouvé en gris.

Références

[1] G. Dewaele, F. Devernay et R. Horaud. Hand Motion from 3D Point Trajectories and a Smooth Surface Model. Article en cours de soumission à *European Conference on Computer Vision*, 2004.

[2] Q. Delamarre et O. Faugeras. Finding Pose of Hand in video images : a stereo based approach. In *Proc. 3rd Int. Con. on Automatic Face and Gesture Recognition*, April 1998, pp. 585-590

[3] J. Rehg et T. Kanade. DigitEyes : Vision-Based Human Hand Tracking. *IEEE Workshop on Motion for Non-Rigid and Articulated Objects*, 1994, pp. 16-22.

[4] R. Plankers et P. Fua. Tracking and Modeling People in Video Sequences.*Computer Vision and Image Understanding*, vol. 81, 2001, pp. 285-302.

[5] W. E. Lorensen et H. E. Cline. Marching cubes : A high resolution 3D surface construction algorithm. In *Proc. of the 14th Conference on Computer Graphics and Interactive Techniques*, 1987, pp. 163-169.

[6] J. F. Blinn. A Generalization Of Algebraic Surface Drawing. *ACM Transactions on Graphics*, 1(3), 1982, pp. 235-256.

[7] F. Le Jeune. Suivi de la main dans une séquence vidéo. *Rapport de stage de DEA*, DEA IFA, Université de Marne-la-Vallée, France, 2001

[8] J.C. Terrillon, M. David et S. Akamatsu. Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments. In *Proc. of the Third International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, 1998, pp. 112-117.

[9] P. Fua. From Multiple Stereo Views to Multiple 3D Surfaces. *International Journal of Computer Vision*, vol. 24, August 1997, pp. 19-35.

[10] W.H Press, S.A. Teukolsky, W.T. Vettering and B.P. Flannery. *Numerical Recipes in C++ : The Art of Scientific Computing, Second Edition*. Cambridge University Press, New York, 2002.

[11] *http ://www.ptgrey.com*

[12] *http ://www.curiouslabs.com*