La main comme télépointeur

Nicolas Roussel, Ghislain Nouvel[*]

Laboratoire de Recherche en Informatique
UMR 8623 CNRS - Université Paris-Sud
LRI - Bât. 490 - Université Paris-Sud
91405 Orsay Cedex, France
{roussel,nouvel}@lri.fr

RESUME
Cet article décrit une nouvelle approche au problème de la communication gestuelle dans les collecticiels. Cette approche est basée sur l'utilisation comme télépointeur de l'image de la main de l'utilisateur filmée en temps réel et détourée par un procédé de chroma-keying. Le résultat de cette technique est que les utilisateurs peuvent communiquer par gestes plus efficacement qu'avec des télépointeurs traditionnels.

MOTS CLES : Collecticiel, Télépointeur, Geste, Superposition d'images, Chroma-keying.

INTRODUCTION
Le geste est un moyen naturel de coordination et de communication entre individus. Les mains, en particulier, sont souvent utilisées pour exprimer des idées, se référer à des objets, attirer l'attention ou faire circuler la parole. Pour rendre possibles ce type d'échanges à distance, les collecticiels synchrones proposent généralement des télépointeurs que les participants peuvent déplacer au-dessus des objets partagés.

Les télépointeurs traditionnels ne sont que de simples curseurs de souris. Ces curseurs sont sémantiquement pauvres : ils ont une taille et une orientation fixe, et leur forme se restreint à un choix dans un ensemble prédéfini. Ces restrictions limitent la communication par gestes entre les participants. Il est par exemple difficile d'attirer l'attention ou de désigner plusieurs objets en même temps. La plupart du temps, les participants sont donc obligés d'ajouter des annotations temporaires (e. g. cercles et flèches) aux objets partagés afin de contourner ces difficultés [3].

Un curseur de souris est un pauvre substitut pour le geste manuel. Les collecticiels synchrones bénéficieraient grandement de télépointeurs offrant une plus grande richesse sémantique. Cette idée n'est pas nouvelle : certains outils collaboratifs modifient déjà la taille ou la forme des télépointeurs et les complètent éventuellement par l'ajout d'informations textuelles en fonction de l'activité des participants [2].Cependant, malgré toutes ces améliorations, il semble que la propre main de l'utilisateur serait le télépointeur idéal, à la fois intuitif et naturel. Dans cet article, nous présentons une technique basée sur le chroma-keying qui permet à un utilisateur de collecticiel d'utiliser l'image de sa propre main comme télépointeur.

LA MAIN COMME TÉLÉPOINTEUR : PRINCIPE
Le chroma-keying est un procédé utilisé en télévision et au cinéma pour insérer un fond (e. g. une carte météorologique ou un paysage) derrière un sujet principal (e. g. le présentateur ou les acteurs). Pour obtenir ce résultat, le sujet est filmé sur un fond de couleur unie, généralement bleu, et les images ainsi obtenues sont traitées pour remplacer les zones de cette couleur clef par les zones correspondantes du fond à insérer.

Nous mis en œuvre cette technique en installant une caméra au-dessus de l'un de nos bureaux. En plaçant sous cette caméra un tissu ou une large feuille de papier coloré, le procédé de chroma-keying nous permet de détourer l'image des mains de l'utilisateur ou de tout autre objet posé sur le fond uni (Figure 1). Ces zones détourées peuvent alors être superposées à l'affichage d'une application. L'intérêt de cette technique est double : les participants peuvent utiliser leurs mains pour communiquer par gestes de façon plus naturelle et intuitive qu'avec un curseur, et ils peuvent également utiliser des objets réels pour annoter la vue.

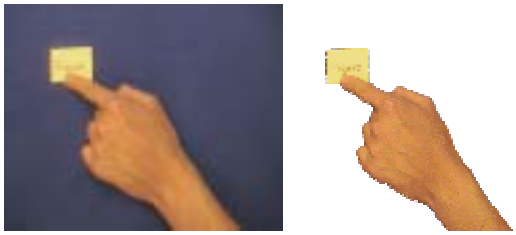

Figure 1 : Avant et après le procédé de chroma-keying

Lorsque l'image de la main apparaît à l'écran, elle est souvent trop grande, ce qui occulte une trop grande

[*] L'adresse actuelle du deuxième auteur est : Ghislain Nouvel, Philips Consumer Communication, User Requirement Specification Group, ghislain.nouvel@dev-lme.pcc.philips.com

partie de l'image de fond et rend la désignation de petits objets difficile. Pour pallier ce problème, l'utilisateur peut régler le niveau de transparence de l'image de la main et voir ainsi à la fois sa main et les objets partagés qui sont en dessous, comme c'est le cas avec TeamWorkstation [4] ou VideoDraw [6]. D'autre part, l'utilisateur peut régler le facteur d'échelle (ou zoom) de l'image de la main et rendre celle-ci arbitrairement petite à l'écran. Cela pose cependant un problème : le champ couvert par la caméra n'autorise plus alors le déplacement de la main sur l'ensemble de l'écran ou de la fenêtre de l'application. Dans nos prototypes, l'utilisateur peut utiliser la souris pour déplacer l'image de la main sur l'ensemble de l'application.

LA MAIN COMME TÉLÉPOINTEUR : PROTOTYPES

Deux prototypes ont été réalisés afin de tester de façon informelle l'utilisabilité d'images vidéo détourées pour le télépointage. Ces prototypes ont été implémentés sur stations SGI Octane et O2 à l'aide de videoSpace [5], une boîte à outils permettant l'intégration de vidéo en temps réel dans des applications. Le processus de chroma-keying opère à environ 15 images 360x288 pixels par seconde selon l'algorithme suivant (implémenté en logiciel).

La couleur clef est calculée comme la moyenne des couleurs présentes dans une image de référence montrant le fond uni seul. Puis, pour chaque pixel des images suivantes, on calcule sa luminosité ainsi qu'une distance entre ses composantes RGB et celles de la couleur clef. Ces deux valeurs nous permettent de calculer le niveau d'opacité (ou *alpha value*) du pixel :

- si le pixel est sombre, l'opacité sera inversement proportionnelle à la distance colorimétrique (cas d'un objet foncé posé sur le fond ou d'une ombre) ;
- si le pixel est clair et suffisamment proche de la couleur clef, l'opacité sera nulle (pixel appartenant au fond) ;
- sinon, l'opacité est fixée au maximum autorisé par l'utilisateur (pixel appartenant à la main ou à un objet).

Ces trois cas permettent de différencier le fond à ignorer, le premier plan et, dans une certaine mesure, les ombres (certains objets foncés posés sur le fond pouvant cependant être mal interprétés). Afin de pouvoir calibrer l'algorithme, deux seuils de tolérance sont utilisés pour déterminer si un pixel est sombre, et s'il est suffisamment proche de la couleur clef.

Simulation sur un fond fixe ou préenregistré

Notre premier prototype utilise la librairie graphique OpenGL pour superposer deux flux vidéo : le premier flux montre ce qui est supposé être la vue partagée, tandis que le second montre la vue de la caméra placée au-dessus du bureau de l'utilisateur. Chaque image provenant de cette caméra est traitée par l'algorithme de chroma-keying décrit ci-dessus. Le résultat est ensuite superposé sur les images de la vue partagée, grâce à la gestion de l'opacité/transparence faisant partie intégrante du modèle graphique d'OpenGL. L'image utilisée comme fond peut être une image fixe comme la capture d'écran de la Figure 2, mais également un flux vidéo.

Figure 2 : Superposition d'images détourées : ici, la vue partagée est une capture d'écran montrant un document HTML

L'utilisateur déplace le flux de vidéo détourée sur la vue partagée en déplaçant la souris, et contrôle la taille en zoomant à l'aide de deux des boutons (Figure 3). Le clavier peut être utilisé pour contrôler les valeurs des deux seuils utilisés pour le chroma-keying (par exemple pour ajuster les seuils prédéfinis à des conditions d'éclairage particulières) ainsi que l'opacité maximale des zones détourées (degré de transparence de la main).

Quelques tests informels montrent que le contrôle à la souris étant identique à celui que l'on exerce sur un curseur traditionnel, l'utilisation du système se révèle rapide et efficace pour les tâches de pointage. En plus des formes habituelles de communication gestuelle, la possibilité d'agrandir ou de diminuer la taille tout en se déplaçant offre de nouvelles possibilités : il est par exemple possible de désigner un groupe d'objets en les "prenant dans sa main" grâce à l'utilisation de la transparence. Cette possibilité est également fort utile pour l'annotation par l'ajout d'objets réels que l'on peut précisément disposer. Une version pilotant le déplacement par détection de mouvement sur le flux vidéo a également été testée, mais s'est révélée moins pratique, à cause de son manque de souplesse.

Figure 3 : Après un changement de taille, de position et du niveau transparence

Ce premier prototype nous a permis de tester l'utilisation d'un flux vidéo comme pointeur et de vérifier l'intérêt de cette démarche. Cependant, la vue partagée étant simulée par un flux vidéo, elle reste limitée à l'utilisation d'une capture d'écran ou de séquences vidéo. Suite à ces premiers essais, nous avons donc conçu une nouvelle application destinée à superposer le flux de vidéo détourée par chroma-keying sur une quelconque application en cours d'utilisation.

Simulation sur une application en cours d'exécution

La plupart des serveurs X-window disponibles aujourd'hui disposent d'un plan graphique spécial appelé *overlay plane* qui permet à une application d'afficher des éléments graphiques au-dessus des fenêtres présentes à l'écran. Ce plan graphique est notamment utilisé pour afficher des menus déroulants ou des rectangles de sélection sans nécessiter le réaffichage des applications temporairement masquées. L'utilisation d'*overlay planes* complique quelque peu le fonctionnement du serveur X, aussi leur profondeur (i.e. le nombre de couleurs disponibles) est-elle généralement limitée à 8 bits au plus (soit 256 couleurs). L'*overlay plane* étant affiché au-dessus des plans graphiques standards, une de ses couleurs est réservée pour indiquer les zones transparentes. Cette possibilité offerte par X-Window pour la gestion de l'opacité des zones détourées par le chroma-keying est malheureusement binaire : contrairement à OpenGL, la semi-transparence est impossible.

Notre second prototype demande à l'utilisateur de sélectionner l'application qui servira de fond (application hôte) en cliquant sur celle-ci. Il en détermine alors la fenêtre principale et crée une nouvelle fenêtre fille de celle-ci dans l'*overlay plane*. Comme dans le premier prototype, les images provenant d'une source vidéo sont traitées par chroma-keying, mais subissent en outre une réduction du nombre de couleurs par *dithering* pour pouvoir être affichées. La semi-transparence étant impossible, on ne distingue que le fond (transparent) et les mains ou objets posés dessus : les ombres sont éliminées. La taille de la nouvelle fenêtre est ajustée pour être identique à celle du flux vidéo superposé.

La gestion des événements liés au clavier et à la souris est plus problématique. Si notre application s'abonne à ces événements, ils ne sont plus transmis à l'application hôte. Il faut donc trouver un compromis entre le contrôle du flux vidéo superposé et le contrôle de l'application hôte. Notre prototype s'abonne aux événements souris pour pouvoir positionner et redimensionner sa fenêtre sur celle de l'application. Il s'abonne également aux événements correspondant au relâchement de touche (KeyRelease) pour le réglage des seuils et de l'opacité, ces événements étant peu souvent utilisés dans les autres applications. En conséquence, l'utilisateur peut interagir normalement avec l'application hôte tant que sa souris n'est pas sur la fenêtre de notre prototype. Dans ce dernier cas, la souris ne peut servir qu'au contrôle du flux vidéo. Cette solution n'est que partiellement satisfaisante et nous poursuivons l'étude d'une meilleure méthode d'intégration.

CONCLUSION ET PERSPECTIVES

Nous avons présenté une nouvelle approche au problème de la communication gestuelle dans les collecticiels basée sur l'utilisation de vidéo détourée par chroma-keying comme télépointeur. Deux prototypes ainsi que différentes simulations nous ont permis de valider cette approche et d'en mesurer les limites. Notre algorithme de chroma-keying peut encore être amélioré, de même que la gestion des entrées de notre second prototype. Cependant, nous pouvons d'ors et déjà faire quelques commentaires sur la technique développée dans cet article.

Tout d'abord, il paraît évident que l'étape suivante est l'intégration de notre technique dans un collecticiel existant pour pouvoir en tester l'utilisabilité avec plusieurs utilisateurs. La boîte à outils videoSpace permettant d'accéder à des sources vidéo aussi bien locales que distantes, les modifications nécessaires pour l'implémentation de réels télépointeurs consistent à transmettre la position, la taille et l'opacité maximale contrôlées par l'utilisateur. Si le collecticiel utilise déjà des télépointeurs, cette intégration devrait se faire sans trop de problème, les mécanismes de transmissions de position étant déjà existants. Pour ce qui est de l'affichage de nos télépointeurs, le premier prototype donne la solution pour les applications utilisant OpenGL et le deuxième en propose une plus générique pouvant être adaptée à toute application X-Window.

On peut penser que lorsque les utilisateurs d'un collecticiel pourront utiliser leurs mains pour désigner et communiquer, ils voudront sans doute aussi pouvoir créer, déplacer ou modifier des objets. Ce passage du geste sémiotique au geste ergotique [1] nécessitera sans aucun doute des traitements d'image plus sophistiqués ou des outils comme des tablettes graphiques. Ces outils pourraient également offrir de nouvelles possibilités pour contrôler le flux vidéo superposé (par exemple contrôler l'orientation des images par des rotations de la souris).

Le télépointeur remplit de nombreuses fonctions dans les collecticiels. En plus de support à la communication gestuelle, il renseigne sur l'activité du groupe en montrant la position des différents utilisateurs et permet de les identifier, généralement par l'utilisation de couleurs. Peut-on reconnaître un utilisateur à ses mains ? Que va-t-on voir s'il tape au clavier ? Sans doute faut-il combiner notre technique avec d'autres existantes ou à inventer.

BIBLIOGRAPHIE

1. C. Cadoz. Le geste canal de communication homme/machine. La communication "instrumentale". Technique et Science Informatique, 13(1):31-61, 1994.
2. S. Greenberg, C. Gutwin, and M. Roseman. Semantic Telepointers for Groupware. In Proc. OzCHI'96, Hamilton, New Zealand, pages 24-27, November 1996.
3. S. Hayne, M. Pendergast, and S. Greenberg. Implementing gesturing with cursors in Group Support Systems. Journal of Management Information Systems, 10(3):43-61, 1994.
4. H. Ishii, M. Kobayashi, and K. Arita. Iterative Design of Seamless Collaboration Media. Communications of the ACM, 37(8):83-97, August 1994.
5. N. Roussel and M. Beaudouin-Lafon. VideoSpace: A Toolkit for Building Mediaspaces. Research report, LRI, Université Paris-Sud, France, May 1999.
6. J.C. Tang and S.L. Minneman. VideoDraw: A Video Interface for Collaborative Drawing. ACM Transactions on Information Systems, 9(2):170-184, April 1991.