Université de Liège
Faculté de Sciences Appliquées
Télécommunications et Imagerie

Année académique 2004-2005

INTERACTION HOMME-MACHINE EN TEMPS RÉEL PAR TRAITEMENT D'IMAGES VIDÉOS

Renaud Dardenne

Mémoire réalisé en vue de l'obtention
du Diplôme d'Études Approfondies
(DEA) en Sciences Appliquées

Table des matières

1 Introduction **5**

1.1 Motivation générale et contexte de la recherche 5

1.2 Description du projet CINEMA 6

1.3 Nos objectifs 6

2 Acquisition vidéo au moyen d'une caméra IEEE 1394 **7**

2.1 Équipement 7

2.1.1 USB 7

2.1.2 FireWire 8

2.1.3 Camera Link 8

2.1.4 Comparaison 9

2.2 Le bus IEEE 1394 10

2.3 Transmission vidéo par IEEE 1394 11

2.4 La partie pratique 14

2.4.1 Configuration des périphériques 14

2.4.2 Librairies (API) nécessaires 15

2.4.3 Un exemple d'acquisition multi-caméras en C 16

2.5 Le problème de l'éclairage 21

3 Détection de la tête et des mains et applications **23**

3.1 Description du problème 23

3.1.1 Le mouvement 23

3.1.2 Détection des zones de peau 25

3.1.3 Critères de suppression des zones non-pertinentes 26
3.2 Reconnaissance de mouvements 27
3.2.1 Utilisation d'une caméra supplémentaire 28
3.2.2 Redéfinition des mouvements 29
3.2.3 Combinaison des informations et résultats 30
3.3 Calcul général de la distance 30
3.3.1 Position du problème 30
3.3.2 État de l'art 32
3.3.3 La méthode employée 32
3.4 Conclusion 35

4 Intégration avec la partie audio 37
4.1 Projet CINEMA : partie audio 37
4.1.1 Ambiance acoustique 37
4.1.2 Spatialisation 38
4.2 Le système audio de CINEMA 38
4.3 Communication 38

5 Résultats 40

6 Conclusions 48

Bibliographie 50

Remerciements 53

Chapitre 1

Introduction

1.1 Motivation générale et contexte de la recherche

De nombreux travaux en réalité virtuelle proposent l'immersion de l'utilisateur par des casques immersifs recréant un espace visuel et sonore complètement disjoint de l'environnement réel dans lequel se trouve l'utilisateur. L'interaction de l'utilisateur avec l'espace virtuel se fait par des capteurs de position de bras, mains et jambes posés à même le corps. Les moyens à mettre en œuvre sont coûteux et individualisés pour chaque utilisateur. C'est pour cette raison qu'une alternative intéressante consiste à capturer les mouvements de l'utilisateur par une ou plusieurs caméras, à segmenter l'utilisateur et lui proposer dans son champ de vision un écran sur lequel il est re-projeté mais intégré dans un décor enrichi par des éléments visuels de synthèse qui réagissent à ses actions.

Dans ce cas le dispositif d'interaction est constitué d'une caméra interfacée à un PC dans lequel une image, fusion de la réalité (la personne en mouvement) et de l'histoire virtuelle est créée et re-projetée sur le mur qui fait face à l'utilisateur (comme un miroir magique qui fait que la personne se voit dans un environnement virtuel). Cette nouvelle modalité d'interaction a déjà été étudiée dans le cadre d'autres projets. Les bases scientifiques de cette première étape étaient constituées par une maîtrise technologique de la segmentation en temps réel d'objets en mouvement et de l'identification des paramètres essentiels de l'objet. De progrès récents en acquisition de la profondeur (que ce soit par stéréovision ou par caméra unique), en composition de scènes 3D de réalité augmentée à partir de cartes de distances rapides et en création d'environnements sonores spatialisés ouvrent de nouvelles perspectives de création d'environnement d'immersion.

A fin de réaliser ce nouvel environnement d'immersion audio-visuelle non-intrusive permettant la mise en œuvre de nouvelles modalités de narration et d'apprentissage, les équipes des professeurs M. Van Droogenbroeck, J.-J. Embrechts, J. Hancq et B. Macq ont été réunies.

1.2 Description du projet CINEMA

Alors que de nombreuses recherches scientifiques ont étés effectuées des derniers temps dans les domaines du traitement de l'image, de la reconnaissance et synthèse de la parole et de la création d'environnement sonores, peu d'efforts ont porté sur l'intégration de ces modalités. Les communication immersives, constituant le domaine d'application de notre approche, sont en plein essor, permettant la création d'environnements de réalité augmentée (l'histoire part de la vie réelle et est enrichie par des éléments de synthèse) dont l'application suscite un engouement très supérieur à la réalité virtuelle. Un élément clef de la démarche proposée dans ce projet réside également dans le concept de compositeur de scène largement étudié dans la cadre de la norme MPEG-4.

L'objectif global du projet est d'arriver au terme de trois années de recherche de fournir un environnement immersif offrant aux utilisateurs de s'immerger, tant par la vue que par l'audition, au sein d'aventures virtuelles à caractère ludique, culturel ou éducatif.

Les objectifs spécifiques sont :

– Immersion des utilisateurs dans des environnements tri-dimensionnels, renforçant l'impression de présence réelle de par l'usage de la profondeur.
– Étude et conception d'avatars animés automatiquement et parlant grâce à de la synthèse vocale qui pourront interagir utilement avec les utilisateurs, que ce soit pour les guider dans un contenu (pédagogique) complexe ou pour jouer avec eux.
– Immersion sonore : développement d'une station de création d'espaces sonores virtuels incluant une base de données de sons anéchoïques et d'outils d'auralisation, de spatialisation et de synthèse vocale.
– Extraction automatique de données tridimensionnelles à propos de l'utilisateur pour renforcer ses capacités d'interaction naturelles avec le système ; extraction de cartes de profondeur résistantes aux changements d'illumination et calcul de paramètres de mouvement et la capture combinée image-parole-son.
– Développement de caméras intelligentes (caméras intégrées avec un processeur DSP) permettant l'intégration efficace des techniques précitées.
– Création d'un environnement intégré et expérimentation au cours d'un événement majeur (MILIA par exemple).

1.3 Nos objectifs

Parmi les objectifs cités, les tâches qui nous étaient originalement assignées étaient la création de cartes de profondeur résistantes aux changements d'illumination, la détection dans le champ 3D et la modélisation des mouvements de l'utilisateur et la composition de scènes hybrides synthétiques naturelles respectant les perspectives 3D.

Au fil de ce rapport, ces objectifs seront modifiés ou adaptés par les conditions du dispositif expérimental.

Le chapitre 2 traitera du matériel que nous avons acquis ainsi que de son utilisation, le chapitre 3 discutera des méthodes employées pour réaliser les différents objectifs, le chapitre 4 parlera brièvement de la partie audio du projet et finalement le chapitre 5 illustrera nos résultats.

Chapitre 2

Acquisition vidéo au moyen d'une caméra IEEE 1394

Ce chapitre est une adaptation d'un article paru dans *Linux Magazine* en février 2005 [15]. Il explique en quoi le choix de nos caméras est important et constitue le point de départ de notre travail.

2.1 Équipement

Nous allons commencer par justifier notre choix de la norme IEEE 1394 pour l'acquisition vidéo. Il existe différentes normes et différents types de bus permettant d'effectuer le transfert d'images, en voici les principales :

2.1.1 USB

L'Universal Serial Bus (USB) est un bus qui permet de connecter des périphériques externes à un ordinateur (hôte dans la littérature USB). Il supporte 127 périphériques simultanés. Le bus supporte les branchements et débranchements à chaud et fournit l'alimentation électrique des périphériques.

La version 1.1 du bus peut communiquer dans deux modes : lent (1,5 Mbits/s) ou rapide (12 Mbits/s).

- Le mode lent permet de connecter des périphériques qui ont besoin de transférer peu de données, comme les claviers et souris.
- Le mode rapide est utilisé pour connecter des imprimantes, scanners, disques durs, graveurs de CD, et autres périphériques ayant besoin de plus de rapidité. Néanmoins il est insuffisant pour beaucoup de périphériques de stockage de masse (par exemple, il ne permet que la vitesse 4x sur les lecteurs/graveurs de CD).
- La nouvelle version de ce bus, USB 2.0, comporte un troisième mode permettant de communiquer à 480 Mbit/s. Il est utilisé par les périphériques rapides : disques durs, graveurs,...

Le bus USB supporte un protocole plug-and-play. Dès la connexion, l'hôte lit certaines informations sur le périphérique. Celles-ci lui permettent d'identifier le périphérique (type, constructeur, nom, version) et donc facilitent le travail du système pour déterminer le driver le plus approprié.

L'hôte communique successivement avec chaque périphérique, le débit total est donc partagé entre l'ensemble des périphériques. Il est possible d'attribuer à certains périphériques un débit constant pour une période de temps. Le reste du débit est toujours attribué de façon équitable entre les autres périphériques.

Une autre caractéristique du protocole USB est la possibilité de structurer la communication entre un hôte et un périphérique en plusieurs canaux logiques pour simplifier la commande du périphérique. Par exemple sur un disque dur USB, il est commode de disposer d'un canal pour passer les commandes et d'un autre séparé pour passer les données.

2.1.2 FireWire

FireWire est le nom d'une norme d'interface série multiplexée, aussi connue sous le nom IEEE 1394, également aussi appelée interface iLink. Il s'agit d'un bus rapide véhiculant à la fois des données et des signaux de commandes des différents appareils qu'il relie.

Ce bus est *Plug and Play*, on peut donc l'utiliser pour brancher toutes sortes de périphériques gourmands en bande passante, notamment disques durs et caméscopes numériques. Elle permet l'alimentation du périphérique, ainsi que le raccordement de 63 périphériques par bus et leur branchement/débranchement à chaud. On peut raccorder jusqu'à 1024 bus par l'intermédiaire de bridges.

FireWire a été inventé par Apple au début des années 1990 et peut atteindre des débits de plusieurs dizaines de MBytes/s. Son objectif clairement affiché était de remplacer à terme le bus USB, en tout cas pour les périphériques par lesquels circulent des flux importants de données.

Le FireWire supporte le *Hot Plug* (branchement à chaud), la connexion ou la déconnexion d'un périphérique déclenche un événement chez tous les autres périphériques (événement bus reset), ainsi tout le monde sait à tout moment qui est présent sur le bus. À chaque bus reset les périphériques reçoivent un numéro d'identification de 0 à n (max 62), celui qui a le plus grand numéro est élu chef du bus ou *root*, c'est lui notamment qui est chargé de marquer le début de cycles de 125 microsecondes jouant le rôle d'ordonnanceur. Tout les périphériques peuvent être *root* ainsi contrairement à l'USB il n'est pas nécessaire de relier les périphériques à un ordinateur pour qu'ils communiquent entre eux.

2.1.3 Camera Link

Camera Link est une méthode de transmission numérique, conçue par des fabricants de vision pour le marché de la vision industrielle. Il s'agissait pour eux de répondre à une demande pressante des utilisateurs qui souhaitaient pouvoir connecter caméras et cartes de traitement de façon simple et interchangeable.

Camera Link est une spécification de câblage. Le câble inclut les informations de transmission de données, mais aussi le contrôle de la caméra, et les communications série asynchrones, le tout

sur un seul câble. Dorénavant, deux câbles suffisent, l'un pour l'alimentation l'autre pour Camera Link. Et cela, quel soit le type de caméras ou de cartes de traitement.

Sur le plan des performances le taux maximum de transferts annoncé est de 2,3 GBits/sec. Le cœur de ce qui se veut un nouveau standard, est un chip mis au point par National Semiconductor. Cette puce est intégrée dans chaque récepteur et chaque transmetteur.

Camera Link utilise la signalisation différentielle de basse tension, ou le LVDS, qui est un système de signalisation électrique pouvant fonctionner à grande vitesse sur des câbles de cuivre torsadés bon marché. Il a été présenté en 1994, et est depuis lors devenu très populaire. LVDS peut transporter les signaux vidéo jusqu'à environ 10m et ce pour des résolutions d'affichage de 1400×1050 (SXGA+) à 60 Hz.

2.1.4 Comparaison

Le tableau 2.1 reprend un résumé des avantages et inconvénients des différentes normes.

	Avantages	Inconvénients
USB	Bas prix Simplicité d'utilisation Paramétrage facile Débit classique de 480 Mbits/s	Bande passante limitée Topologies limitées (étoile)
Firewire	La bande passante est garantie Les périphériques ne nécessitent pas d'alimentation supplémentaire Transport d'information sur de grandes distances Compatibilité avec les futures versions de la norme Existence d'un standard de communication Possibilité d'effectuer du preprocessing Débit classique actuel de 400 - 800 Mbits/s	Nécessite un driver respectant le standard de communication
Camera Link	Grande (énorme) bande passante Débit classique de 2.3 Gbits/s	Nécessite une carte d'acquisition dédiée Communication unidirectionnelle Pas de protocole Le prix !

TAB. 2.1 – Comparaison des différentes normes d'interfacage

Au vu de ce tableau, nous avons porté notre choix sur la norme IEEE 1394 ; en effet, ses avantages sont multiples : il existe un driver Linux, la bande passante garantie est suffisante, le bus peut

fournir suffisamment de courant, elles ne sont pas trop onéreuses et l'existence d'un standard de communication vidéo nous assure la possibilité de communiquer directement avec la caméra.

2.2 Le bus IEEE 1394

Fondamentalement, la norme IEEE 1394 n'est rien de plus que la description d'un bus. Mais, comme il s'agit d'un bus ayant une vitesse typique de 400 Mbit/s, la description prend plus que quelques pages.

Le débit du bus IEEE 1394 est fonction de la variante de la norme que l'on considère. Le tableau 2.2 reprend les différentes variantes.

Norme	Débit théorique
IEEE 1394a-S100	100 Mbit/s
IEEE 1394a-S200	200 Mbit/s
IEEE 1394a-S400	400 Mbit/s
IEEE 1394b-S800	800 Mbit/s
IEEE 1394b-S1200	1200 Mbit/s
IEEE 1394b-S1600	1600 Mbit/s
IEEE 1394b-S3200	3200 Mbit/s

TAB. 2.2 – Débits théoriques des variantes de la norme IEEE1394.

Parmi toutes ces normes, la norme IEEE 1394a-S400 est la plus répandue, ce qui correspond à un débit maximum de 400 Mbit par seconde.

Le bus 1394, dont l'architecture est reproduite à la figure 2.1, peut fonctionner selon 2 modes distincts :

– *asynchrone*. Ce mode de transfert est basé sur une transmission de paquets à intervalles de temps variables. Cela signifie que l'hôte envoie un paquet de données et attend de recevoir un accusé de réception du périphérique. Si l'hôte reçoit un accusé de réception, il envoie le paquet de données suivant, sinon le paquet est transmis à nouveau après un certain délai.
– *isochrone*. Ce mode permet l'envoi de paquets de données de taille fixe à intervalle de temps régulier (cadencé grâce à deux fils d'horloge). De cette façon, aucun accusé de réception n'est nécessaire, ce qui garantit un débit fixe et donc une bande passante. De plus, étant donné qu'aucun accusé n'est nécessaire, l'adressage des périphériques est simplifié et la bande passante économisée permet de gagner en vitesse de transfert. Les communications isochrones sont prioritaires par rapport aux asynchrones ce qui permet de garantit la bande passante pour des applications du type transmission vidéo temps-réel ; ce qui n'est pas le cas des transactions asynchrones transitant par la couche transaction. Si le débit est théoriquement égal à 400 Mbit/s, le cadencement des paquets d'une transmission isochrone fait que le débit utile ne dépassera pas 256 Mbit/s. En pratique, on est donc loin de la vitesse promise !

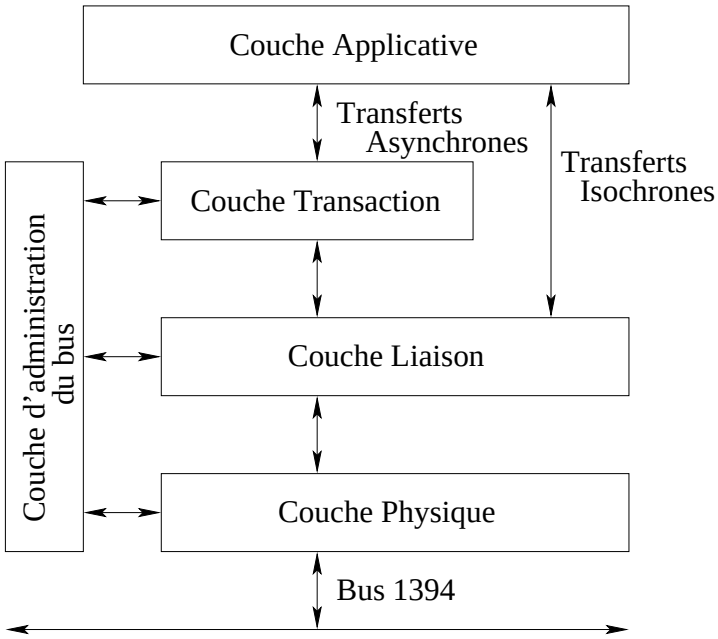

FIG. 2.1 – Architecture des protocoles IEEE 1394.

2.3 Transmission vidéo par IEEE 1394

La transmission de signaux vidéo n'est pas explicitement couverte par des normes de la famille IEEE 1394. C'est le groupement d'industriels appelé *1394 Trade Association* qui s'est chargé de définir une norme spécifiquement dédicacée à la transmission vidéo. C'est précisément le sous-groupe *Instrumentation & Industrial Digital Camera* (IIDC) qui a développé un protocole pour la transmission de signaux de caméras ; il s'agit du protocole *Digital Camera* (DCAM). Au passage, notons que la plupart des auteurs font indifféremment référence à l'IIDC et la norme DCAM pour qualifier le protocole de transmission. DCAM prévoit :

1. la transmission asynchrone d'informations de paramétrisation des caméras (taille, système de couleur, luminosité, balance des blancs, etc).
2. la transmission isochrone de flux vidéo *non comprimés*.

Il existe également un format appelé DV (pour *Digital Video*) à ne pas confondre avec les formats DVCAM et DVPRO. Ce format est utilisé notamment par les caméscopes numériques à interface IEEE 1394 mais, à la grande différence avec DCAM, il définit le transfert de flux vidéo *comprimés*. Comprimer ou ne pas comprimer ? La polémique continue de faire rage entre constructeurs.

Nous nous contenterons de décrire l'acquisition suivant le protocole DCAM ; sans compression donc. Examinons donc de plus près le nombre de caméras utilisables sur un même bus.

Prenons par exemple une caméra capable de fournir 30 images au format progressif, en couleurs (sur 24 bits) et de taille 640x480. Par caméra, cela représente un débit (en bits) de 30x24x640x480, soit 221 Mbit/s.

	RGB	YUV (4 :2 :2)	YUV (4 :1 :1)	Monochrome (4 :2 :2)
Taille de l'image	640x480	640x480	640x480	640x480
Octet par pixel	3	2	1,5	1
Nombre d'images par seconde	30	30	30	30
Débit	221 Mbit/s	147 Mbit/s	111 Mbit/s	74 Mbit/s

TAB. 2.3 – Calcul explicite de débits pour certains formats vidéo.

A priori, il n'est donc pas possible d'utiliser 2 caméras couleurs sur une carte FireWire ; en tous cas à la cadence nominale de 30 images par seconde. Mais c'est oublier un peu vite que les formats vidéo sont multiples. En effet, le format *RVB* (*RGB* en anglais), qui consiste à attribuer un octet par composante de couleur rouge, verte ou bleue par pixel, n'est pas la représentation la plus efficace. Il se fait que le système visuel humain est plus sensible au signal de luminance qu'aux signaux de couleurs. Comme le système RVB n'en tire aucun profit, certains préfèrent utiliser le système *YUV* (ou *YCbCr*) qui a l'avantage de séparer une composante moyenne des signaux RVB, appelée *luminance* (Y), de deux signaux de couleurs (U, V ou Cb, Cr) appelés *chrominances*. Le mérite de ce changement de représentation est avant tout de permettre une représentation des chrominances avec moins d'échantillons que celle de la composante Y car l'œil y est moins sensible. Cela a conduit à définir une série de systèmes YUV en fonction de l'échantillonnage des composantes YUV, dont voici les principaux :

4 :4 :4 Ce schéma préserve la grille d'échantillonnage originale ; il y a donc un échantillon de luminance et deux échantillons de chrominance par pixel.

4 :2 :2 C'est le format obtenu en conservant toutes les lignes mais seulement un échantillon de chrominance sur 2 par ligne. Autrement dit, par échantillon de luminance, il y a soit un échantillon U, soit un échantillon V. Cela représente une moyenne de 2 octets par pixel, contrairement au schéma 4 :4 :4 qui nécessite 3 octets par pixel.

4 :1 :1 Basé sur le même principe d'échantillonnage, on ne garde ici qu'un échantillon par chrominance sur 4. Autrement dit, pour deux échantillons de luminance, il y a, au total, un échantillon de chrominance. Le débit total est donc 1,5 celui de la luminance.

Un logiciel comme Coriander [3] permet de se familiariser avec la panoplie de paramètres ajustables par DCAM (voir illustration 2.2).

Nous sommes maintenant en mesure de calculer les débits pour quelques formats vidéo numériques typiques. Ces débits sont repris dans le tableau 2.3.

Le débit maximum typique d'une carte FireWire est limité à 400 Mbit/s. Hélas, un fonctionnement isochrone du bus –indispensable pour des transmissions continues de signaux vidéo– limite le débit utile à 256 Mbit/s, ce qui signifie qu'il n'est pas possible de brancher 2 caméras au format YUV 4 :2 :2 sur une même carte sans réduire le nombre d'images par seconde. Bien entendu, rien n'empêche de brancher plusieurs caméras sur différentes cartes FireWire ; ainsi, 2 cartes permettront d'acquérir simultanément les signaux de plusieurs caméras, même à haut débit. Notons au passage qu'avec des caméras monochromes, il est possible d'accroître leur nombre ou

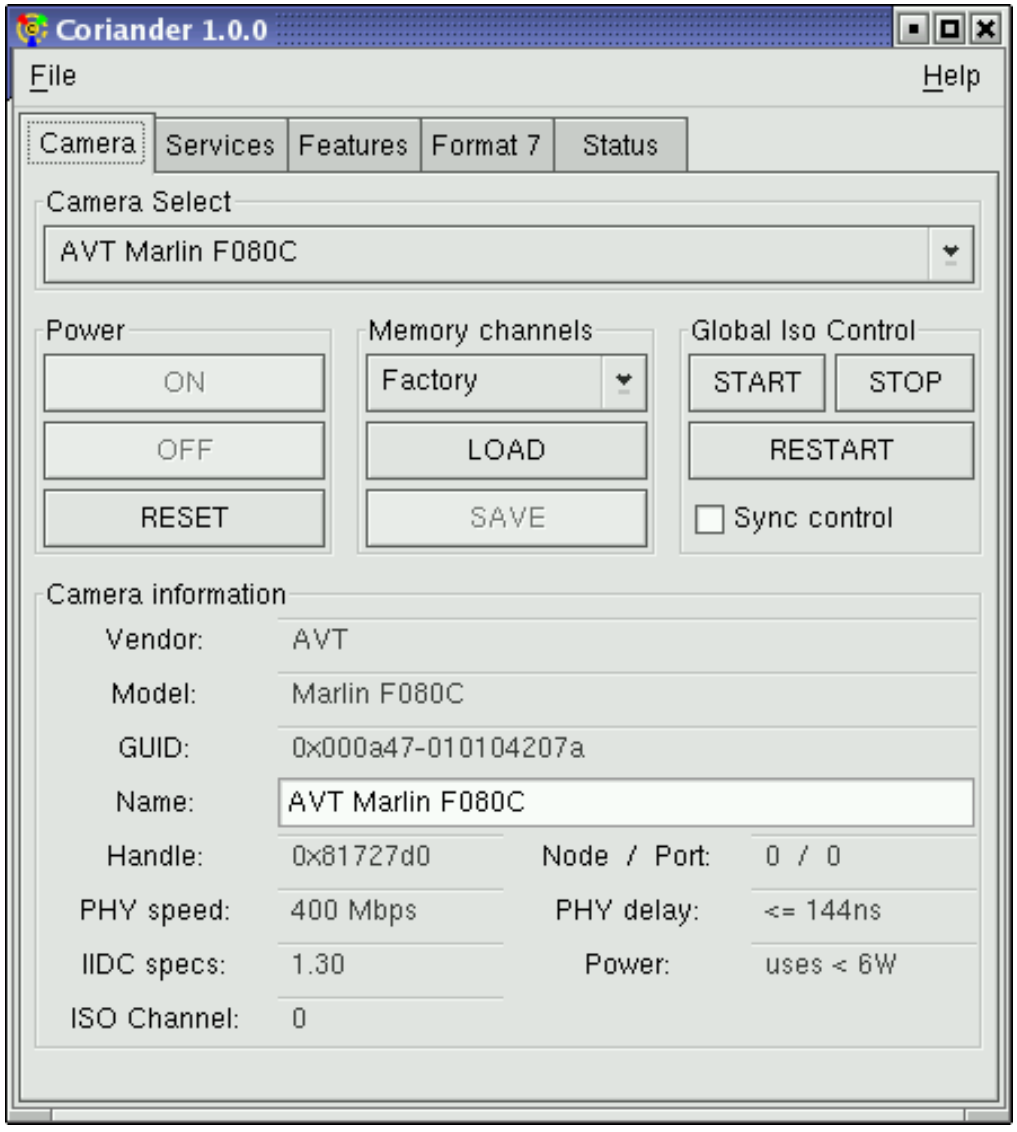

FIG. 2.2 – Fenêtre du logiciel Coriander.

d'augmenter la taille de l'image, ce qui peut convenir pour des applications industrielles plus exigeantes.

Des améliorations (IEEE 1394b) ont été prévues pour porter la bande passante de 400 Mbit/s à 800, 1600 et même à 3200 Mbit/s. Signalons tout de même que ces débits seront limités par la bande passante du bus du processeur. En effet, le bus PCI est limité à 1 Gbit/s. Il faudra dès lors attendre une généralisation de ses successeurs (PCI-X, PCI-Express) avant de pouvoir pleinement bénéficier de ces capacités.

2.4 La partie pratique

Cette fois, nous entrons dans le vif du sujet. Pour l'acquisition, il faut configurer les périphériques, installer des librairies et écrire un peu de code. Voici comment procéder. Cette section, contrairement aux suivantes, contient volontairement le détail du code car il n'existe pas à l'heure actuelle de documentation sur l'acquisition IEEE1394 sous Linux et il nous a semblé justifié de rentrer dans le détail de cette partie de notre travail.

2.4.1 Configuration des périphériques

Cette section décrit ce dont le système a besoin pour pouvoir acquérir en toute sérénité :

1. Tout d'abord disposer d'une carte 1394 (!). Il en existe de deux types : celles respectant la norme OHCI et celles ne la respectant pas. Inutile de dire que les cartes appartenant à la première catégorie sont mieux supportées sous linux. On peut facilement en vérifier la présence à l'aide de `/sbin/lspci -vvx`.
2. La présence d'un noyau récent (`uname -r`) ne peut qu'éviter des écueils : au minimum supérieur à la version 2.4.20, un noyau 2.6 étant plus que conseillé pour éviter de nombreux problèmes d'instabilité du système et profiter des dernières améliorations parmi lesquelles l'amélioration du sous-système d'entrée/sortie, du sous-système des modules et périphérique unifié ainsi que le support du Hot Plug. La principale modification qui nous intéresse consiste en une réécriture du support Firewire à l'aide du système Hot Plug : puisque la connexion à chaud est la règle de nos jours et non plus l'exception, la nouvelle infrastructure de périphériques élimine principalement les différences entre un périphérique ancien, et un connectable à chaud. Comme le sous-système du noyau ne différencie pas directement un périphérique découvert au lancement d'un périphérique branché a posteriori, la plupart de l'infrastructure pour gérer les périphériques connectables a été simplifiée.
3. Ensuite, il faut vérifier la présence des périphériques adéquats dans `/dev` : `/dev/video1394` et `/dev/raw1394` doivent tous deux être présents et posséder respectivement les numéros de majeurs/mineurs suivants : (171/0) et (171/16). Attention : ces valeurs ne sont correctes que sur des systèmes dont le noyau est supérieur à 2.4.20. Voici, dans le cas d'un noyau 2.6.4 les entrées dans `/dev` d'un système sur lequel il est prévu de brancher quatre caméras :

```
crw-rw-rw- 1 root 171, 0 Sep 15 2003 /dev/raw1394
crw-rw-rw- 1 root 171, 16 Sep 15 2003 /dev/video1394/0
```

```
crw-rw-rw- 1 root 171, 17 Sep 15 2003 /dev/video1394/1
crw-rw-rw- 1 root 171, 18 Sep 15 2003 /dev/video1394/2
crw-rw-rw- 1 root 171, 19 Sep 15 2003 /dev/video1394/3
```

Certains systèmes paranoïaques empêchent la lecture par un utilisateur de ces périphériques. Il convient donc d'ajuster les permissions.

4. Les drivers 1394 sont habituellement chargés en tant que modules. Il faut en vérifier le chargement correct : les modules requis sont `ieee1394`, `ohci1394`, `raw1394` et `video1394`. Le système doit normalement charger de lui-même les deux premiers (à vérifier avec la commande `dmesg`) si une carte est présente. `raw1394` et `video1394` doivent quant à eux être chargés à la main. Il suffit d'ajouter les lignes suivantes à la fin du fichier `/etc/rc.d/rc.local`

   ```
   /sbin/modprobe raw1394
   /sbin/modprobe video1394
   ```

   et de vérifier que tout s'est bien passé (à l'aide de lsmod). Que sont les modules `raw1394` et `video1394` ? Ce sont des drivers de haut niveau. Les fonctions de raw1394 sont appelées lorsqu'un hôte est ajouté ou supprimé, lorsque les drivers de bas niveaux envoient des données en mode isochrone, etc. `video1394` implémente un accès matériel via un canal DMA pour le réception et l'envoi de données isochrones, ce qui signifie que le processeur ne doit pas prendre en charge les opérations de copie des données vers la mémoire centrale.

Un éventuel problème peut survenir en conséquence du chargement du module eth1394 au démarrage, empêchant le chargement des autres drivers. `eth1394` est un module qui, comme son nom l'indique (!), permet le transport de datagrammes IPv4 sur un bus IEEE1394. Cette possibilité a été décrite dans le Request For Comments (RFC) numéro 2734. Notons toutefois que le module ne supporte pas le protocole complet.

Lorsque toutes ces opérations ont été réalisées. On peut vérifier que tout s'est bien passé en branchant les caméras et en lançant l'utilitaire `systool`, faisant partie du package `sysfsutils`, qui permet de donner une foule de renseignements sur la structure et le contenu du système de fichier virtuel `/sys` (uniquement noyaux 2.6) reflétant la configuration réelle de la machine.

2.4.2 Librairies (API) nécessaires

Il existe des librairies correspondantes aux 2 drivers de haut niveau que sont raw1394 et video1394 : libraw1394 [10] et libdc1394 [9]. Toutes deux sont disponibles sur `sourceforge`, mais elles sont toujours en développement.

La librairie libraw1394 (version 1.1.0) inclut tout ce qu'il faut pour récupérer les données brutes via le bus IEEE 1394 et même pour la norme IEEE 1394b à 800 Mbit/s ! La librairie libdc1394 est indispensable pour ne pas consommer de ressources CPU (ce qui est possible grâce à l'accès DMA) et, comme elle est mise à jour régulièrement, il convient de télécharger la dernière version (1.0.0). Remarquons que l'installation des librairies ne fait pas toujours correctement appel à `ldconfig` ; il est donc parfois nécessaire de mettre à jour soi-même le cache des librairies pour l'édition de liens.

2.4.3 Un exemple d'acquisition multi-caméras en C

Pour cet exemple, nous avons choisi une acquisition multi caméras car cette approche est plus générique sans pour autant augmenter de manière significative la complexité du code. Les instructions comportent les phases suivantes.

2.4.3.1 Inclure les librairies

Sans quoi rien ne fonctionnera :

```
#include <libraw1394/raw1394.h>
#include <libdc1394/dc1394_control.h>
```

2.4.3.2 Vérifier que tout est prêt

Même si après la lecture des précédents paragraphes, tout devrait être opérationnel, nous allons tout de même vérifier le bon chargement des modules. Puisque libdc1394 est une implémentation plus efficace de l'acquisition, on peut se demander pourquoi libraw1394 est nécessaire. La réponse est que la librairie définit un type que nous allons recroiser par la suite : `raw1394handle_t`. Nous allons commencer par vérifier que les drivers sont bien chargés et que nous avons les permissions suffisantes pour accéder aux périphériques en lecture :

```
// Déclaration pour raw1493
raw1394handle_t handle;
struct raw1394_portinfo ports[MAX_PORTS];

// Assigner un "handle"
handle = raw1394_new_handle();
if (handle==NULL) {
  fprintf( stderr, "Unable to aquire a raw1394 handle\n\n"
     "Please check \n"
     " - if modules 'ieee1394','raw1394' and 'ohci1394' are
        loaded \n"
     " - if you have read/write access to /dev/raw1394\n\n");
  exit(1);
}
handle = NULL;
```

2.4.3.3 Compter le nombre de ports

Une fois cette vérification d'usage effectuée, nous savons que tout ce qui est nécessaire à l'acquisition est en place. Nous allons poursuivre en vérifiant le nombre de cartes FireWire présentes sur la machine ; dans la terminologie FireWire, une carte est appelée "port" :

```
// Déclarations pour raw1394
raw1394handle_t handle;
struct raw1394_portinfo ports[MAX_PORTS];
int numPorts;

// Assigner un "handle"
printf("Asking new handle ...\n");
handle = raw1394_new_handle();
if (handle==NULL) {
  fprintf( stderr, "Unable to aquire a raw1394 handle\n\n");
  exit(1);
}

// Compter le nombre de ports
printf("Getting the number of ports ...\n");
numPorts = raw1394_get_port_info(handle, ports, numPorts);
printf("[ %2d ]",numPorts);
raw1394_destroy_handle(handle);
handle = NULL;
```

où `MAXPORT` est une macro spécifiant le nombre maximal de cartes potentiellement connectées à la machine.

2.4.3.4 Compter le nombre de nœuds sur chaque port

Par pure commodité nous supposerons que certaines variables sont globales :

```
nodeid_t *camera_nodes;
raw1394handle_t handles[MAX_CAMERAS];
dc1394_cameracapture cameras[MAX_CAMERAS];
int camCount=0;
```

`camera_nodes` est une structure qui nous permet de compter les nombre de caméras effectivement connectées, `handles` est un tableau de descripteurs que nous utiliserons sans cesse, `cameras` qui est du type `dc1394_cameracapture` contient toutes les données sur les caméras y compris le buffer de données associé et `camCount` sera le nombre de caméras effectivement connectées.

```
int numPorts;
int p=0;
raw1394handle_t handle;
int found=0;

// Pour chaque port
for (p = 0; p < numPorts; p++) {
```

```
  // Demander un handle
  handle = raw1394_new_handle();
  raw1394_set_port(handle, p );

  // Demander le nombre de caméras connectées
  camera_nodes = dc1394_get_camera_nodes(handle, &camCount, 1);

  // Libérer le handle
  raw1394_destroy_handle(handle);
  handle = NULL;
  found += camCount;
}
// found contient le nombre de caméras conectées
```

L'appel à la fonction `dc1394_get_camera_nodes` permet de remplir la structure `nodeid_t *camera_nodes` avec les données propres à chaque caméra qui seront utilisées lors de l'initialisation de l'acquisition, ainsi que de montrer une description de la caméra trouvée.

2.4.3.5 Fixer les paramètres de l'acquisition

C'est, en nombre de lignes de code, la partie la plus longue ; c'est ici que nous allons fixer les paramètres comme la taille des images, le nombre d'images par seconde, l'espace de couleurs utilisé, etc. Avant de se lancer dans le code, nous allons faire un petit rappel des différents formats existants. Le standard IIDC 1.30 définit 7 formats se distinguant par la résolution maximale que chacun peut atteindre. Chacun de ces formats se subdivise en modes spécifiant une résolution et un espace de couleur précis. Chacun de ces modes se subdivise à son tour afin de spécifier le nombre d'images par seconde qui peut être compris entre 1.875 et 60 image/s. Bien sûr, toutes les caméras ne supportent pas tous les modes à tous les débits. Il faut se référer au manuel du matériel pour connaître les modes supportés. La librairie libdc1394 comporte bien des appels (cf. `dc1394_control.h`) permettant de demander aux caméras les modes qu'elles supportent, mais nous ne les utiliserons pas pour ne pas surcharger le code. La description des premiers modes est fournie dans le tableau 2.4.

```
#define DROP_FRAMES 0

// Définition des débits et des modes supportés par toutes les
   caméras
int fps = FRAMERATE_30; // 30 images/sec
int res = MODE_640x480_MONO;
int formatGeneral = FORMAT_VGA_NONCOMPRESSED;
char *device_name=NULL;

// iso transmission
unsigned int channel;
unsigned int speed;
```

Mode	Description
0	160x120 YUV (4 :4 :4)
1	320x240 YUV (4 :2 :2)
2	640x480 YUV (4 :1 :1)
3	640x480 YUV (4 :2 :2)
4	640x480 RGB
5	640x480 Y (Mono)
6	640x480 Y (Mono16)

Format 0

Mode	Description
0	800x600 YUV (4 :2 :2)
1	800x600 RGB
2	800x600 Y (Mono)
3	1024x768 (4 :2 :2)
4	1024x768 (RGB)
5	1024x768 Y (Mono)
6	800x600 Y (Mono16)
7	1024x768 Y (Mono16)

Format 1

TAB. 2.4 – Description des premiers modes des formats 0 et 1.

```
int i, p;
// Pour chaque port
for (p = 0; p < portCount; p++) {
  // Pour chaque caméra
  for (i = 0; i < thisCamCount; i++) {
    handles[numCameras] = dc1394_create_handle(p);
    if (handles[numCameras]==NULL)
      perror("Unable to aquire a raw1394 handle\n");
    // Obtenir un canal ISO
    if (dc1394_get_iso_channel_and_speed(handles[numCameras],
        cameras[numCameras].node,&channel, &speed) !=
            DC1394_SUCCESS) {
      printf("unable to get the iso channel number\n");
      cleanBus();
      exit(-1);
    }
    // Fixer les parametres
    if (dc1394_dma_setup_capture(handles[numCameras],
        cameras[numCameras].node, i+1, formatGeneral, res,
            SPEED_400, fps,
        NUM_BUFFERS ,DROP_FRAMES, device_name, &cameras[
            numCameras])
          != DC1394_SUCCESS) {
      perror("Unable to setup capture");
      cleanBus();
      exit(-1);
    }

    // Commencer la transmission ISO
    if (dc1394_start_iso_transmission(handles[numCameras],
```

```
        cameras[numCameras].node) != DC1394_SUCCESS) {
      perror("unable to start camera iso transmission\n");
      cleanBus();
      exit(-1);
    }
  }
  dc1394_free_camera_nodes(camera_nodes);
}
```

Il faut évidemment veiller à ce que votre caméra supporte le bon nombre d'images par secondes (*framerate*) à la résolution demandée.

Pour une description complète des appels aux différentes fonctions de la librairie, il est nécessaire de lire les commentaires qui se trouvent dans le fichier `dc1394_control.h` de libdc1394. Il n'existe malheureusement aucune documentation officielle à ce jour ...

2.4.3.6 Acquérir

Une fois tous les paramètres fixés aux bonnes valeurs, acquérir est une simple formalité :

```
// Acquérir
dc1394_dma_multi_capture(cameras, camCount);

// cameras[x].capture_buffer contient les données
// Placez vos traitements ici.
[...]

// Rendre le buffer au driver
for (i = 0; i < camCount; i++)
  dc1394_dma_done_with_buffer(&cameras[i]);
```

Notez qu'avant de réutiliser le buffer, il faut signaler qu'on a fini de l'utiliser. Parmi les traitements à effectuer, il faut très souvent commencer par une conversion d'espace de couleurs. Des routines de conversion efficaces de YUV vers RGB, par exemple, sont faciles à trouver : Coriander en possède pour les cas les plus fréquents.

2.4.3.7 Une gestion propre de la sortie

Lorsque pour une raison ou pour une autre un appel a échoué, un simple appel à `exit()` est un peu barbare. C'est dans cette section que se justifie le plus la déclaration globale des variables `handles`, `cameras` et `camCount` : nous commençons par arrêter la transmission isochrone, nous arrêtons les caméras, nous libérons le canal DMA et les handlers.

```
void cleanBus()
{
```

```
  int i;
  printf("\n Stopping ISO transmissions");
  for (i=0; i < camCount; i++)
    if (dc1394_stop_iso_transmission(handles[i], cameras[i].node
        )
          != DC1394_SUCCESS) {
      perror(" *** unable to stop camera iso transmission ***");
      exit(-1);
    }

  printf("\n Cameras PowerOff");
  for(i=0 ; i< camCount; i++)
    if (dc1394_camera_off(handles[i], cameras[i].node) !=
       DC1394_SUCCESS)
      perror(" *** PowerOff Failed ***");

  printf("\n Releasing DMA channels and freeing handlers");
  for (i=0; i < camCount; i++) {
    dc1394_dma_unlisten( handles[i], &cameras[i] );
    dc1394_dma_release_camera( handles[i], &cameras[i]);
    dc1394_destroy_handle(handles[i]);
  }
}
```

Cette fonction sans arguments peut être aussi positionnée sur le signal de Ctrl+C.

2.4.3.8 Afficher l'image

Cette section sort un peu du cadre propre de l'acquisition IEEE 1394, mais une visualisation des images issues des caméras peut être réalisée très aisément à l'aide de GTK2 ou de SDL.

GTK est sous licence GNU LGPL et permet de créer très rapidement des interfaces graphiques. GTK+ existe sur plusieurs plates-formes : plates-formes UNIX-like, Windows, BeOs. GTK+ est également utilisable avec plusieurs langages de programmation.

SDL permet de réaliser des programmes qui utilisent des images, des sons, les lecteur de CD-ROM, qui communiquent en réseau etc. La bibliothèque est très populaire auprès des applications multimédia, en outre les jeux vidéo, les démos et les émulateurs. Elle peut être également utilisée avec plusieurs langages de programmation, comme le C, le Perl, etc.

2.5 Le problème de l'éclairage

Les caméras que nous avons achetées sont des AVT Marlin F-080-C [4]. Ces caméras permettent de réaliser une acquisition à 15 images par seconde à une résolution de 1024×768 ou une

acquisition à 30 images seconde dans une résolution de 640×480. Ces deux modes sont ceux que nous utilisons le plus souvent.

Des néons classiques fonctionnent à $50Hz$. Lorsque nous prenons une succession d'images, nous échantillonnons les néons ; nos caméras fonctionnant à 15 ou $30Hz$, les néons fonctionnant à une fréquence de $50Hz$, nous échantillonnons la lumière émise par les néons à une cadence inférieure à la fréquence de NYQUIST. Il se produit dès lors un phénomène de repli de spectre qui se traduit par un effet de variation de l'intensité lumineuse entre images successives.

La solution envisagé pour éviter cet effet est l'usage de néons fonctionnant à une fréquence beaucoup plus élevée. En conséquence, lors de l'acquisition d'une image, le temps d'intégration de la lumière par les caméras est suffisant pour comprendre un grand nombre d'oscillations et c'est cet effet de moyenne qui nous permet d'obtenir une intensité lumineuse constante entre images. Ces néons ont de plus l'avantage de ne pas retomber à une intensité nulle entre deux décharges, celles-ci étant trop rapprochées pour laisser au gaz le temps de se désexciter complètement.

Nous sommes maintenant capable de traiter les images acquises par les caméras dans les meilleures conditions.

Chapitre 3

Détection de la tête et des mains et applications

Maintenant que tout est prêt, nous allons pouvoir rentrer dans le vif du sujet. Les objectifs premiers de notre travail sont la détection de la tête et des mains de l'utilisateur. Ces informations seront ultérieurement utilisées pour reconnaître certains mouvements qui permettront d'interagir et de commander la partie audio du projet et pour calculer la distance de la personne par rapport à la caméra.

3.1 Description du problème

Nous allons décrire dans les sections suivantes les différentes étapes préalables à la détection des mains et de la tête de l'utilisateur. Nous passerons en revue les différents modules et nous expliquerons également l'historique des méthodes envisagées pour justifier nos choix. Nous verrons ainsi que certains détails pratiques ainsi que le dispositif expérimental nous ont imposé certains restrictions.

3.1.1 Le mouvement

Nous faisons l'hypothèse que la caméra est fixe –hypothèse justifiée dans le cadre de l'application visée– l'image est donc constituée d'un arrière plan fixe au-dessus duquel vient se superposer une série d'objets dont les personnes, leur visage et leurs mains font partie. La première étape consiste donc à essayer d'obtenir une segmentation de l'avant-plan ou du moins une bonne approximation de celle-ci.

Le calcul de cet arrière-plan (complémentaire de l'avant-plan) nous permet d'exploiter l'information temporelle contenue dans le flux d'images. C'est lors de cette étape que nous prenons en fait que les images successives sont fortement corrélées puisqu'elles représentent toutes l'évolution temporelle d'une même scène.

Cette détection de mouvement se calcule généralement par soustraction de l'image courante avec un arrière-plan calculé [32] [21]. Les différentes méthode se distinguent les unes des autres par la manière dont elles calculent cet arrière-plan. Nous avons implémenté différents algorithmes simples –car nous avons toujours en mémoire la contrainte temps réel– que nous avons testés (simple différence d'images, moyenne pondérée de l'arrière-plan, médiane des précédentes images, etc). La plupart d'entre-eux se sont révélés inefficaces ou présentaient des cas dans lesquels ils ne présentaient pas le comportement espéré comme montré sur la figure 3.1 où apparaissent des phénomène de fantômes et où seuls les contours de la personne sont détectés en mouvement.

Puisque nous avons dû changer l'éclairage du laboratoire dans lequel se trouve le setup expérimental (section 2.5), nous pouvons émettre l'hypothèse supplémentaire que l'éclairage est constant. La plupart des algorithmes de détection de mouvement mettent à jour leur arrière-plan car il se produit des changements dans l'éclairage de la scène (lumière naturelle) ou du bruit. La figure 3.2 montre le résultat de la détection de mouvement à l'aide d'un arrière-plan fixe. Cette méthode simple a ses avantages et inconvénients : les avantages sont une détection parfaite des contours ainsi qu'une détection parfaite d'une personne immobile (en effet dans les autres algorithmes une personne qui ne bouge pas finit toujours par être assimilée à l'arrière-plan). L'inconvénient de la méthode est une mauvaise détection si pour une raison ou pour une autre la caméra vient à être déplacée ou si la lumière naturelle entrant dans la pièce est fortement modifiée.

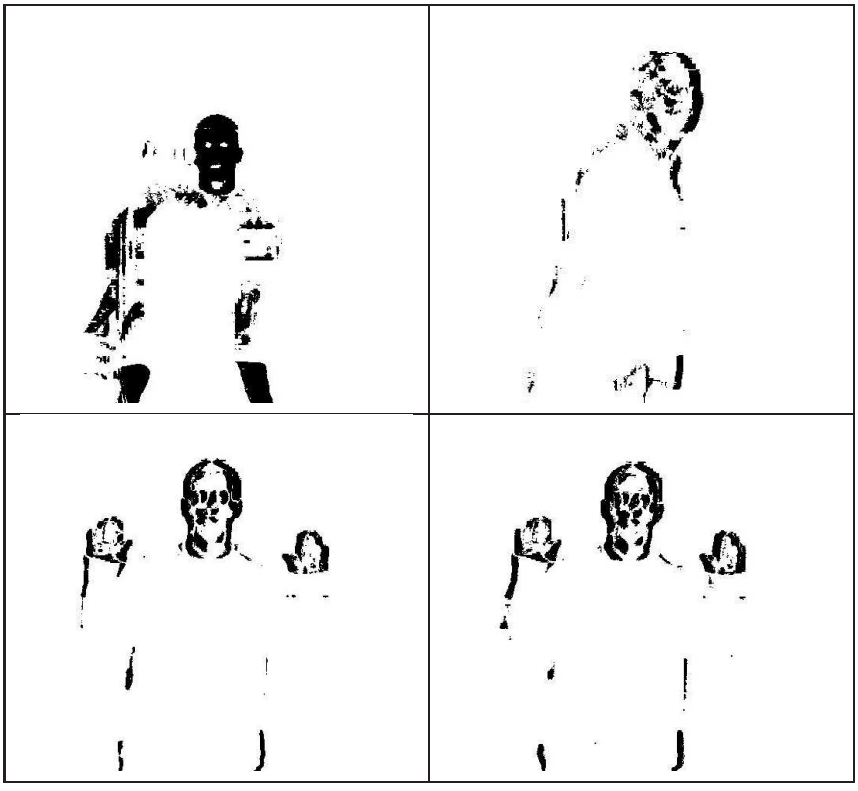

FIG. 3.1 – Mauvaises détections de mouvement.

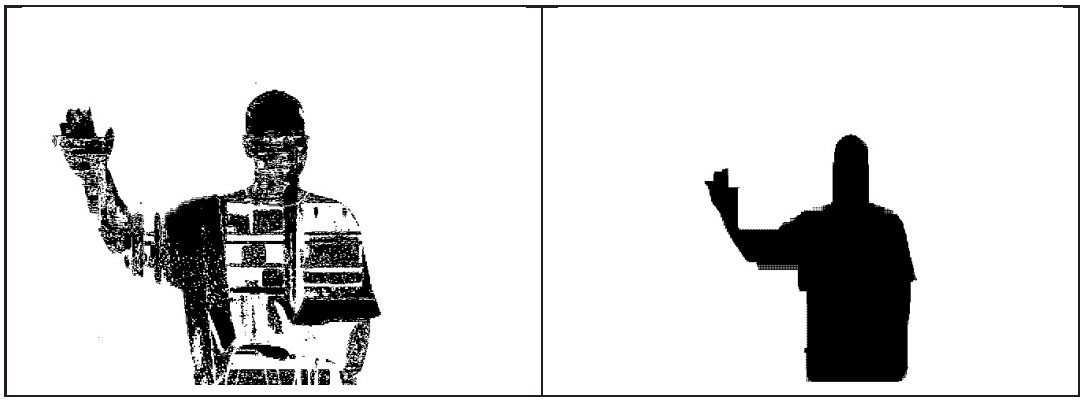

FIG. 3.2 – Détections de mouvement correctes avant et après filtrage morphologique.

3.1.2 Détection des zones de peau

La détection de la couleur de la peau est un moyen efficace souvent utilisé pour définir une série de zones candidates susceptibles de contenir un visage ou des mains. De plus, celle-ci peut être effectuée de manière assez simple grâce à des seuillages dans un espace de couleurs approprié. Une telle détection est alors relativement peu coûteuse en temps de calcul et donc particulièrement adaptée à un détecteur devant traiter un nombre significatif d'images par seconde. Notre deuxième étape consiste donc à détecter les pixels qui sont de la couleur de la peau.

La peau possède des caractéristiques très particulières, quels que soient le type et la couleur de peau, et ce à condition de choisir un espace de couleur suffisamment discriminant. Des travaux [37] ont pu montrer que pratiquement tous les espaces de couleurs donnaient des résultats équivalents (sRGB, YUV, HSV). Il nous faut choisir un espace de couleurs approprié : les espaces convenant le mieux sont ceux qui décorrèlent l'information de luminance de l'information de chrominance. Parmi ceux-ci YUV et HSV sont les deux espaces qui fournissent les meilleurs résultats, néanmoins l'utilisation de l'espace HSV nous permet légèrement d'obtenir de meilleurs résultats mais est beaucoup plus lourd en temps de calcul. Les caméras que nous utilisons fournissent des images appartenant à l'espace YUV. Cet espace séparant déjà les informations de luminance et de chrominance, nous l'avons donc utilisé pour réaliser le seuillage afin d'éviter de coûteuses conversions d'espaces ; la chrominance de la peau chez les êtres humains de tous types étant contenue dans une échelle de valeurs Cb et Cr relativement restreinte (nous utilisons indifféremment les acronymes YUV et YCbCr car ils diffèrent seulement d'un facteur multiplicatif éventuel).

A priori, la combinaison des deux informations de mouvement et de couleur de peau devrait nous permettre de détecter facilement la tête et les mains de l'utilisateur. Nous avons empiriquement constaté que ce n'était pas le cas. En effet, les scènes comportaient des éléments ayant une couleur proche de celle de la peau (meubles, etc) qui conduisaient à des cas de faux positif. De plus, le mouvement détecté n'était pas toujours assez précis. En effet, des ombres étaient détectées comme étant du mouvement. Une ombre projetée sur un élément du mobilier dont la couleur est proche de celle de la peau pouvait donc être détectée.

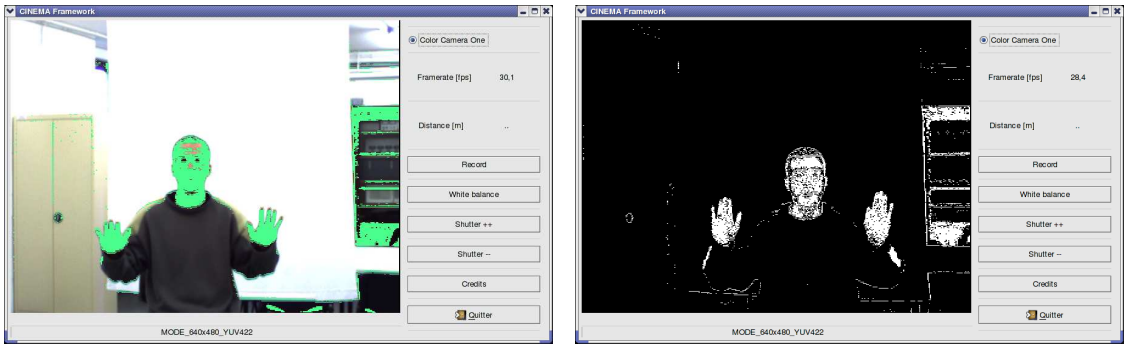

FIG. 3.3 – La détection de peau dans les espaces de couleurs YUV et HSV.

Nous avons implémenté une technique de détection d'ombres inspirée de [12]. Cette technique utilise l'espace HSV pour parvenir à distinguer les objets en mouvement de leur ombre. En effet, dans cet espace de couleur, les valeurs des pixels d'une surface avec et sans ombre ne différent que très peu dans les composantes de saturation et de luminance. Cette méthode est efficace mais elle consomme des ressources en termes d'opérations de calcul, malgré l'utilisation d'une LUT (Look Up Table) effectuant les conversions d'espaces. On peut voir sur la figure 3.4 que la bande blanche verticale indique bien le centre du mouvement réel. Ceci nous permet de ne pas avoir à traiter ultérieurement ces zones, de plus nous nous servirons du centre du mouvement par la suite afin de distinguer la tête des mains.

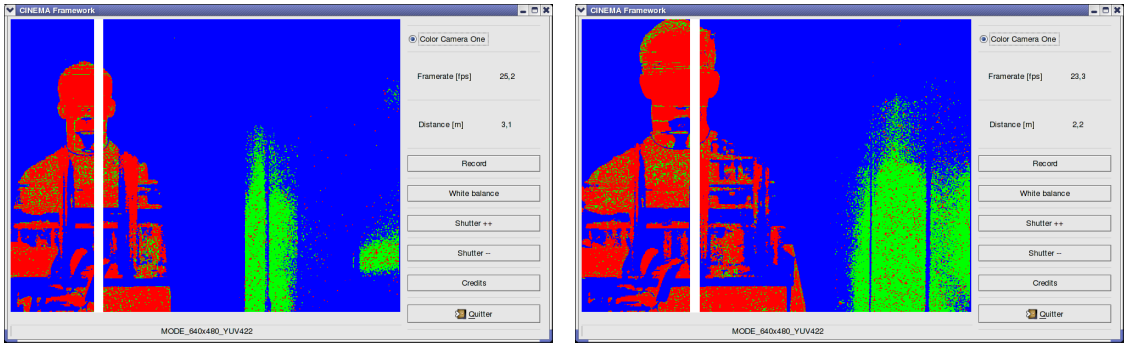

FIG. 3.4 – Le problème de l'ombre.

3.1.3 Critères de suppression des zones non-pertinentes

Une fois ces traitements réalisés, ce que nous obtenons n'est pas suffisant. Il faut soustraire les fausses détections ainsi que distinguer la tête des mains. Il nous faut grouper en zones connexes les pixels d'intérêts et déterminer si ces surface sont elliptiques ou non ainsi que leur orientation principale.

À ce point, nous disposons d'une série de régions d'intérêt parmi lesquelles se trouvent la tête et les mains. Afin de différencier les "bonnes" des "mauvaises", nous ne gardons que les régions

respectant certains critères géométriques de forme, de rapport largeur/hauteur, de remplissage et de connexité (figure 3.5).

Parmi les différentes régions, nous ne conservons que les régions dont la surface est supérieure à une fraction de la plus grande surface connexe, en effet, vus de face la tête et les mains possèdent des tailles relativement proches. Dans l'hypothèse où il n'y a qu'une seule personne dans la scène, nous pouvons ne garder que les 3 plus grandes régions.

Pour distinguer la tête des mains, nous supposons que la tête est la région d'intérêt qui, lorsque nous parcourons l'image depuis sa partie supérieure, est la plus centrale par rapport au mouvement.

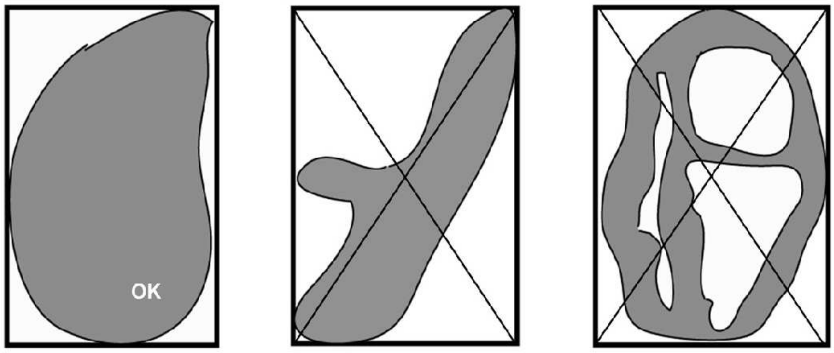

FIG. 3.5 – Exemples de formes et détermination de la conformité ou non aux critères.

Une fois que nous sommes parvenus à distinguer le visage des mains, nous effectuons encore quelques traitements basés sur les mêmes critères de surface afin d'obtenir l'ouverture de la main. Une main est considérée comme fermée si le rapport de la surface de peau sur la surface de la zone rectangulaire la délimitant est supérieur à un seuil.

3.2 Reconnaissance de mouvements

Puisque nous disposons des coordonnées des régions d'intérêt que sont la tête et les mains, nous pouvons utiliser ces valeurs pour effectuer une reconnaissance de mouvement qui nous permettra de commander la partie audio du projet.

Il existe beaucoup de familles d'algorithmes permettant la reconnaissance de mouvements (réseaux de neurones [24], chaînes de Markov cachées [19], etc). Malheureusement toutes ces méthodes sont coûteuses en temps de calcul ou compliquées à implémenter. Nous avons donc opté pour une approche simple. Ceci nous a amené à redéfinir les mouvement de l'utilisateur. Ceux-ci sont extrêmement simples (main ouverte ou fermée, position des deux mains au dessus de la tête) ; ils sont néanmoins suffisamment spécifiques pour être identifiés de manière univoque.

Les mouvements à reconnaître tels que définis initialement dans la convention entre les partenaires du projet sont :

1. La désignation d'une direction vers un instrument virtuel avec un seul bras ;

2. La désignation d'une direction vers un instrument à l'aide des deux bras pour déplacer celui-ci ;
3. Les deux bras levés ;
4. Le traçage d'un arc de cercle avec un bras.

Ces mouvements ont étés définis afin de pouvoir interagir avec le monde virtuel et de commander les différents instruments virtuels présents. Le scénario prévoyait que l'utilisateur soit une sorte de chef d'orchestre dirigeant différents instruments qui peuvent être déplacés dans différentes ambiances sonores. Nous verrons par la suite pourquoi ces mouvements ont étés redéfinis.

3.2.1 Utilisation d'une caméra supplémentaire

Afin de reconnaître les mouvements précités, il apparaît clairement que la seule caméra frontale que nous utilisons n'est pas suffisante. Il nous faut une autre vue pour nous permettre notamment de déterminer la position du bras. Nous avons du donc fixer une deuxième caméra au plafond pour visualiser la personne du dessus. Comme nous l'avons expliqué au chapitre 2, il n'est pas possible de brancher une autre caméra sur le même port car le débit est trop important est le bus est saturé. Nous avons donc installé une carte supplémentaire et nous acquérons une image provenant de cette caméra.

Nous commençons par détecter le mouvement à l'aide de la même technique que pour la caméra frontale (figure 3.6)

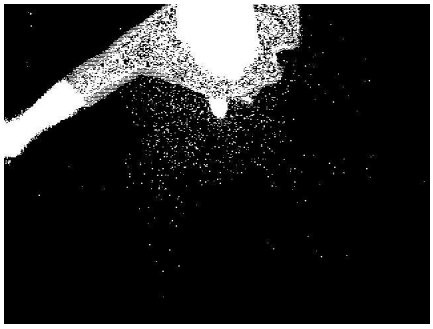

FIG. 3.6 – Caméra plafond : détection de l'avant-plan.

Ce qui nous permet d'obtenir une zone d'intérêt (figure 3.7)

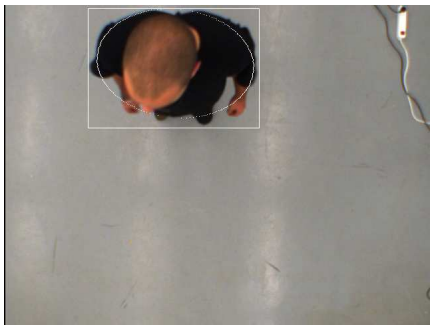

FIG. 3.7 – Caméra plafond : Détection de la zone d'intérêt.

Nous éliminons le mouvement issu de l'ombre de l'utilisateur à l'aide de la même technique que pour la caméra frontale, et nous ne gardons que la plus grande surface.

Ensuite, nous ajustons la meilleure ellipse sur la détection de mouvement afin d'obtenir la direction du bras. Les paramètres de cette ellipse nous fournissent toutes les caractéristiques dont nous avons besoin : l'angle du grand axe (c.-à-d. la direction pointée par le bras) et le rapport des deux axes (l'ellipsoïtude) qui nous permet de déterminer si la personne tend un bras ou non (figure 3.8).

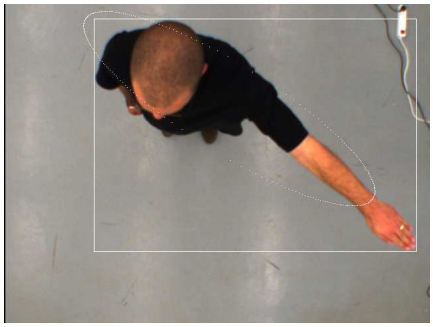

FIG. 3.8 – Caméra plafond : fitting d'une ellipse sur la personne.

3.2.2 Redéfinition des mouvements

Puisque nous approchons le problème de manière simple, une des premières difficultés rencontrées est de redéfinir les mouvements de façon à :

– ce qu'il n'y ait pas de fausses détection positives (il vaut mieux manquer une détection que d'en détecter une alors qu'il n'y en a pas)
– pouvoir déterminer quand un mouvement ou un position devant mener au déclenchement d'un événement commence et quand il se termine.

Pour ce faire nous allons commencer par définir les événements que nous aimerions déclencher :

1. Changer la réponse impulsionnelle de la salle virtuelle dans laquelle l'utilisateur est projeté ;

2. déplacer une source ;
3. modifier le volume d'une source ;
4. rendre toutes les sources muettes.

3.2.3 Combinaison des informations et résultats

Nous n'avons désormais plus qu'à combiner les informations dont nous disposons afin de réaliser une interface permettant de piloter une source sonore. La source ne sera déplacée que dans une seule dimension : en effet nous ne pourrons pas modifier l'élévation de la source puisque nous ne disposons pas d'un suffisamment grand nombre d'enceintes, et les deux dimensions restantes seront modifiées à l'aide de deux mécanismes différents : la distance de la source sera réglée à l'aide du réglage de volume et l'angle sera modifié via la caméra du plafond. Afin de pouvoir tester la bonne reconnaissance des mouvements ainsi que les événements associés nous avons implémenté un rendu visuel très simple (cfr chapitre 5).

3.3 Calcul général de la distance

Le calcul de distance ou de profondeur est employé dans beaucoup d'applications telles que la réalité augmentée ou les interactions homme-machine. La plupart du temps, on considère que le résultat de ce calcul doit fournir une carte dense pour laquelle l'information de profondeur doit être calculée pour chaque pixel. Ces techniques exigent souvent de gros temps de calcul ou des capteurs additionnels. En outre, ces cartes de profondeur denses ne se justifient pas pour toutes les applications. Dans notre travail, nous allons nous limiter à l'évaluation temps-réel de la distance moyenne entre une personne et la caméra. Cette technique, est basée sur la taille apparente de la tête de la personne se trouvant face à la caméra.

3.3.1 Position du problème

L'intérêt pour le calcul d'informations de profondeur pour le suivi de personnes en temps réel a augmenté durant les dernières années. La plupart des techniques sont basées sur des caméras stéréos et ont pour but de fournir des cartes de disparité denses. Malgré les améliorations apportées aux algorithmes, beaucoup de méthodes souffrent toujours de nombreuses déficiences comme la difficulté de détecter de nouveaux objets, la confusion avant-plan / arrière-plan dans les séquences lentes, la mauvaise sensibilité aux changements d'illumination ou l'incapacité de traiter correctement les ombres ou les occlusions.

Une solution qui puisse répondre à ces questions consiste à utiliser les caractéristiques du corps humain ainsi que son comportement. Bjørnstrup [13] a notamment effectué des mesures de segments du corps humain. Ces perfectionnements ont un prix : si l'on veut garder le coût de ces calculs à un niveau acceptable, il faut se contenter de cartes de profondeurs peu denses.

Nous allons donc nous contenter du calcul d'une seule valeur de profondeur représentant la distance séparant l'utilisateur de la caméra. Cette technique est basée sur des données anthropomorphiques (la taille de la tête d'une personne) depuis lesquelles nous dériverons la distance en nous inspirant de la technique du *visual looming* [34].

L'évaluation des paramètres du corps humain n'est pas un sujet nouveau : l'homme vitruvien de De Vinci (figure 3.9) était l'une des descriptions les plus vénérées de composition humaine durant la Renaissance (comme rappelé par le *Da Vinci Code* de Dan Brown). De nos jours, on ne recherche plus une description harmonisant les relations entre les différentes parties du corps, l'évaluation a d'autres buts moins artistiques : l'analyse du mouvement, l'identification des geste ou la conception de prothèse. Drillis et Contini [16] ont étés les premiers à répertorier la taille des segments du corps humain. Leur intérêt initial était la conception de prosthèses améliorées, ils ont ainsi eu besoin de bonnes évaluations de la masse, de centre de la masse et des moments d'inertie de ces segments. Ils ont examiné 20 jeunes sujets masculins pour établir des références de tailles des proportions humaines.

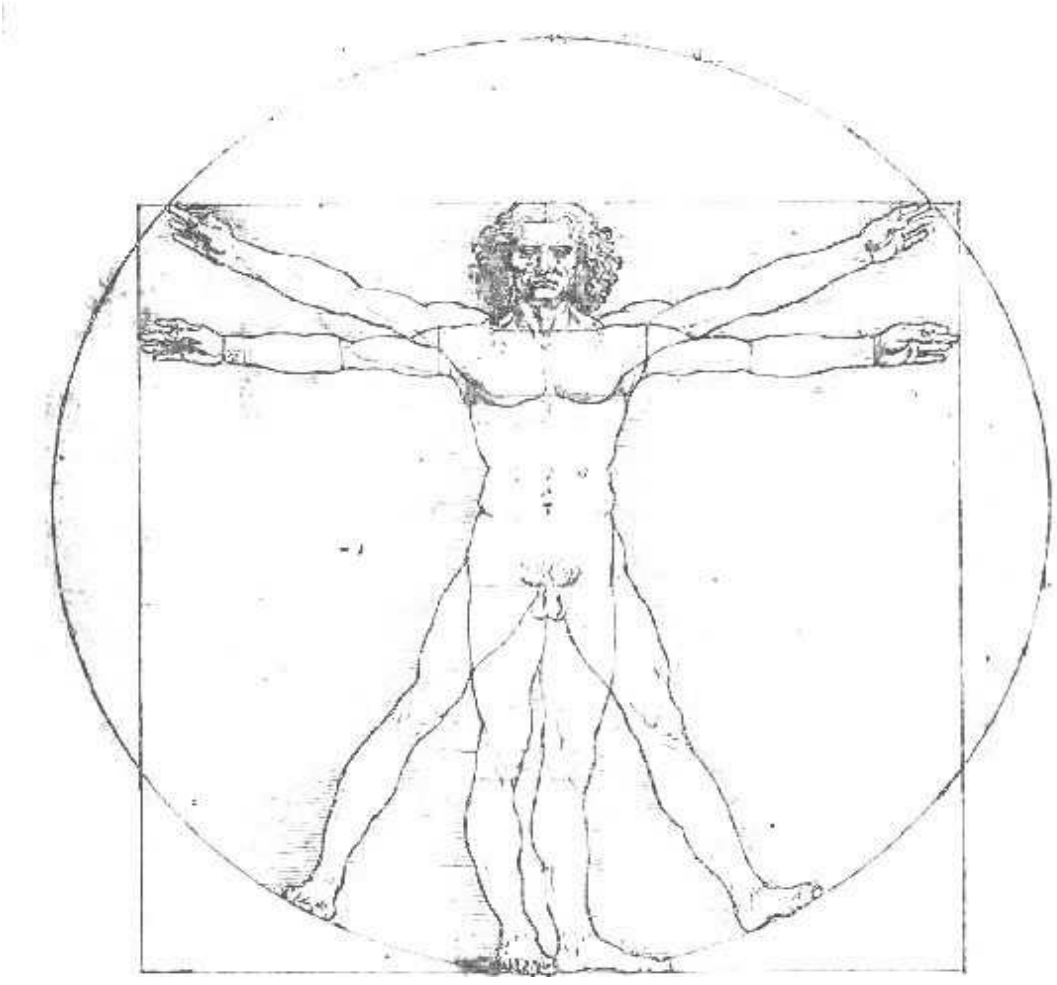

FIG. 3.9 – l'homme vitruvien de Leonard De Vinci.

Dans notre application, nous devons localiser une personne dans une séquence d'images 2D couleur tout en respectant la contrainte du traitement en temps réel. La difficulté majeure de cette tâche est de devoir inférer une information 3D à partir d'une seule caméra, l'information de profondeur étant perdue lors la projection des données 3D sur le plan 2D. Mais une fois que nous avons limité ce problème général à la détection des personnes, nous pouvons employer les

données anthropomorphiques pour fournir l'estimation de profondeur requise.

3.3.2 État de l'art

Il existe d'innombrables techniques pour effectuer le calcul de profondeur [36, 38]. La plupart d'entre-elles ne sont pas adaptées aux hypothèses ou aux besoins de notre application :

– *shape from shading* nécessite des surfaces Lambertiennes –une surface Lambertienne émet ou réfléchit un rayonnement dans toutes les directions avec une radiance (luminance énergétique) constante– or la peau et les vêtements ne sont pas Lambertiens ;
– s*hape from stereo* est une technique beaucoup trop coûteuse en temps de calcul ;
– *shape from texture* et *texture gradient* requièrent la présence de textures. Malheureusement nous n'avons aucune garantie que la scène en contiendra ; ces techniques peuvent également utiliser des systèmes actifs (pinceaux lasers) ;
– *shape from (de)focus* nécessite des caméras à focales variables ;
– *foreshortening* ne fonctionne pas lorsque les objets considérés sont perpendiculaires au plan de la caméra ;
– *motion parallax* suppose que les objets sont en mouvement.

L'élimination de ces techniques nous a amené à développer une technique simple basée sur la technique du *viusal looming* [34].

3.3.3 La méthode employée

Il y a une demande croissante dans les applications biométriques pour l'usage de techniques d'imagerie. La détection de visages et l'identification est l'une de ces applications [40]. La majorité des techniques d'identification utilisent des images 2D en niveaux de gris ou en couleur[17].

Notre méthode combine les étapes suivantes :

1. Obtenir une taille de référence à une distance connue.
2. Détecter la tête de manière continue en combinant la segmentation du mouvement et la détection de peau.
3. Calculer les paramètres de la région d'intérêt détectée et en inférer la distance à la caméra.

Le point 2 ayant déjà été discuté à la section 3.1, nous allons détailler les deux autre points.

3.3.3.1 Obtenir une référence et calculer la distance

Comme nous l'avons déjà dit, notre méthode est inspirée des travaux de [34]. Le principe de cette technique est basée sur les relations entre les déplacements de l'objet par rapport à la caméra et le changement de la taille de la projection de l'objet sur le plan focal de la caméra. La figure 3.10 en montre le principe.

Considérons que la caméra est située en O et que sa distance focale est f. La caméra observe un objet de taille *constante* et *inconnue* h à deux distances D_b et D_a. Nous pouvons remarquer qu'il

est inutile de savoir si le déplacement entre les deux positions est le résultat du mouvement de la caméra ou de l'objet.

À l'aide de triangles semblables, nous pouvons établir l'équation 3.1 :

$$f \times h = P_a \times D_a = P_b \times D_b = k \tag{3.1}$$

où P_a et P_b sont les tailles de l'objet, respectivement situé en D_A et D_b, projetées sur le plan focal, et où k est une constante (cfr figure 3.10).

De cette relation nous pouvons observer :

– qu'elle peut être appliquée dans les directions horizontales ou verticales de la projection ;
– qu'elle suppose implicitement que l'objet se déplace sur une surface plane et perpendiculairement au sol ;
– que puisque notre objectif est de calculer D_b à partir de P_b, il nous suffit d'obtenir une mesure de référence pour être en mesure d'effectuer le calcul.

Bien qu'il n'y ait aucune difficulté particulière à mesurer des distances, nous devons cependant toujours calculer la valeur de k, ce qui implique que nous devons connaître la taille initiale de l'objet dans le plan focal et sa distance originale depuis la caméra. Heureusement il y existe des solutions alternatives à cette première étape de calibrage. Dans [35] il a été proposé de déterminer les paramètres de l'équation en corrélant la réponse de multiples filtres orientés.

Nous exposerons plus tard une autre méthode qui contourne l'étape de calibrage en employant les caractéristiques intrinsèques de la caméra.

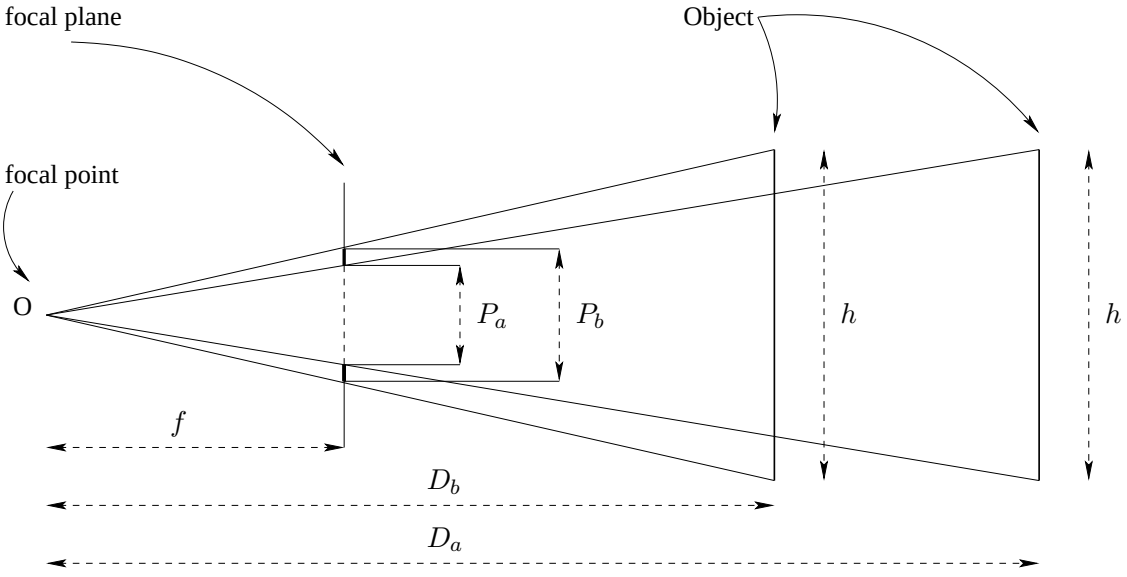

FIG. 3.10 – Géométrie de la technique du Visual Looming.

Distance Réelle	Mesure de H	Mesure de V	Distance estimée de H	Distance estimée de V
7 m	150 pixels	224 pixels	-	-
6 m	175 pixels	261 pixels	6 m	6 m
5 m	211 pixels	313 pixels	4.97 m	5 m
4 m	260 pixels	390 pixels	4.03 m	4.02 m
3 m	346 pixels	513 pixels	3.03 m	3.05 m
2 m	484 pixels	740 pixels	2.16 m	2.11 m

Table 3.1: Mesures obtenues avec un objet rectangulaire.

3.3.3.2 Résultats

Une fois la tête détectée, nous pouvons calculer la distance moyenne de la personne à la caméra. Pour faciliter l'estimation de notre approche, nous avons étudié un objet simple : un rectangle de 120×80 cm. Le tableau 3.1 reprend les résultats obtenus avec cet objet (H et V dénotent respectivement la taille horizontale et verticale de l'objet). La distance de référence utilisée pour calculer k (équation 3.1) est de 7 mètres - distance à laquelle les projections sur le plan focal sont de 150 et 224 pixels. Chacun des deux produits nous amène à deux constantes permettant d'estimer la distance dans les deux directions.

3.3.3.3 Sensibilité de la méthode

Différents facteurs affectent la qualité des résultats :

1. La précision des mesures : La source principale d'erreur provient des incertitudes résultant de l'algorithme de détection de la tête. Une solution consiste à filtrer le résultat de la détection dans les images successives. Malheureusement, lever ou baisser légèrement la tête mène à une mauvaise distance estimée. C'est pourquoi nous avons utilisé dans les tests réels, la largeur du visage plutôt que la hauteur car elle est moins sensibles aux courtes variations. Une autre solution serait d'utiliser un objet plus grand à la place du visage, mais l'équation exige que l'objet soit entièrement visible dans l'image ; l'utilisation du corps entier s'est donc avérée impossible puisqu'on ne peut garantir la visibilité entière du corps dans l'image lorsque le sujet se rapproche de l'objectif.
2. la qualité de l'estimation de la distance est entièrement basée sur la qualité de l'estimation de k. Il est donc impossible de calculer une distance correcte si k a été mal estimé sans intervention humaine.
3. l'effet de perspective des grands objets proches de la caméra influe sur la qualité du résultat car les objets apparaissent déformés. On peut voir, dans la table 3.1, qu'à faible distance, (lorsque la précision sur P_b est grande) l'erreur absolue augmente. C'est bien dû à l'effet de perspective qui déforme le rectangle en parallélogramme.

Une erreur sur k a un impact direct sur la mesure globale de la qualité et consiste en une importante limitation. L'équation du *Looming* nous offre une solution à ce problème.

3.3.3.4 Alternative à l'obtention de la référence

Dans l'équation $f \times h = P_a \times D_a = P_b \times D_b = k$ nous avons calculé k à partir de $P_a \times D_a$; nous pouvons cependant utiliser $f \times h$. Pour ce faire, les spécifications de la caméra nous suffisent.

Dans notre cas, nous utilisons des caméras CCD à une résolution de 1024×768 pixels, une distance focale de 6 mm et une cellule photosensible de 6 mm de diagonale. L'objet de référence a une taille physique h (1200 mm pour la table). Avec ces valeurs, l'équation 3.1 nous fournit une distance D :

$$D = \frac{f \times h}{P_{mm}} \tag{3.2}$$

où P_{mm} est la taille de l'objet dans le plan focal exprimée en *millimètres*. Notre algorithme ne nous fournit pas la taille de l'objet en mm, mais en pixels. Il nous faut donc obtenir une équation similaire à l'équation 3.2 dans laquelle la taille projetée de l'objet est exprimée en pixels.

Puisque le rapport de taille du capteur est $\frac{1024}{768} = \frac{4}{3}$, la connaissance de la diagonale nous permet de calculer les dimensions du capteur rectangulaire (4.8 mm par 3.6 mm). Si h correspond à la hauteur connue de l'objet détecté, l'équation de conversion entre la taille exprimée en mm et la même taille en pixels est $P_{mm} = P_{pixel} \times \frac{3.6}{768}$. Ce qui nous mène à :

$$D = \frac{f \times h}{P_{mm}} = \frac{f \times h}{P_{pixel} \times \frac{3.6}{768}} \tag{3.3}$$

Nous obtenons ainsi une équation similaire pour la largeur w connue de l'objet :

$$D = \frac{f \times w}{P_{mm}} = \frac{f \times w}{P_{pixel} \times \frac{4.8}{1024}} \tag{3.4}$$

S'il existe une manière de mesurer la taille réelle de l'objet considéré dans une direction, il est possible de calculer la distance moyenne de cet objet à la caméra pour peu que cet objet soit pleinement visible par la caméra et que sa projection exprimée en pixels soit bien estimée, ce qui peut dépendre de son orientation. Les équations 3.3 et 3.4 établissent qu'une incertitude relative sur l'estimation de la taille de l'objet est traduit en la même incertitude relative dans le calcul de sa distance D. Il nous faut donc considérer les objets les plus grands possibles.

C'est ici que des données anthropomorphiques peuvent être utilisées : il est possible d'estimer la hauteur de la tête d'une personne à partir de sa hauteur totale (comme montré sur la figure 3.11), la méthode essaie donc tout d'abord d'obtenir la taille de la personne et infère les dimensions de son visage.

3.4 Conclusion

Nous sommes donc parvenu à l'aide de méthodes simples à obtenir tous les paramètres dont nous avions besoin afin de réaliser l'interaction avec la partie audio du projet. Nous connaissons

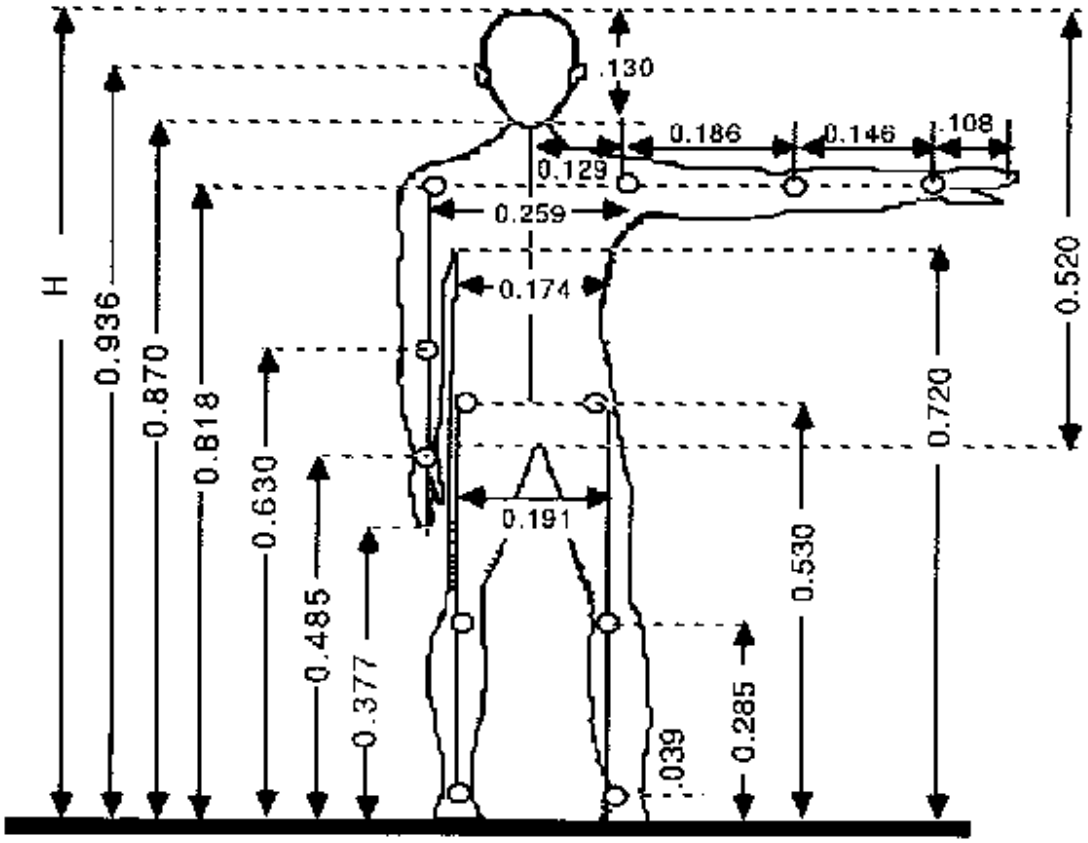

FIG. 3.11 – Rapports entre différents segments du corps humain (depuis [16]).

la distance de la personne à la caméra, nous pouvons reconnaître certains mouvements simples, et grâce à la segmentation nous pouvons effectuer une incrustation de la personne dans un décor virtuel 2D. De plus amples résultats seront présentés au chapitre 5.

Chapitre 4

Intégration avec la partie audio

4.1 Projet CINEMA : partie audio

On peut considérer qu'il y a deux objectifs pour la partie audio de CINEMA :

– Le premier consiste à simuler l'ambiance acoustique de la salle virtuelle. Dans le cadre de CINEMA, en pratique, cette simulation est réalisée par le programme de tirs de rayons *salrev*.
– L'autre objectif est d'assurer aux oreilles de l'utilisateur une reproduction fidèle du champ sonore (= auralisation). Pour obtenir un résultat crédible, le rendu sonore doit non seulement être de bonne qualité au niveau de la sonorité (bonnes caractéristiques fréquentielles, bon rapport signal-bruit, etc), mais également permettre à l'utilisateur de localiser chaque source sonore et de spatialiser l'ambiance acoustique simulée (= localisation et spatialisation).

4.1.1 Ambiance acoustique

Les sons, quel que soit le milieu dans lequel ils se propagent, subissent des réflexions sur les solides qui les entourent (sol, murs). Ces réflexions sont fonction de nombreux paramètres : la forme et la matière des parois réfléchissantes, l'épaisseur de ces parois, l'emplacement de la source dans la pièce, et d'autres paramètres comme le niveau sonore de la source. L'ensemble des réflexions dans ce milieu s'appelle la réverbération.

La réverbération acoustique se divise en plusieurs phases : tout d'abord, les premières réflexions sont provoquées par le retour de l'onde source après sa première réflexion sur les parois et les quelques réflexions suivantes. Les réflexions tardives, plus diffuses, arrivent dans un deuxième temps, après deux ou plusieurs réflexions. Ces deux phases sont bien distinctes et apportent chacune une information différente à l'auditeur.

Les techniques d'auralisation sont assez bien maîtrisées aujourd'hui, à condition que le traitement du signal soit effectué en temps différé. L'auralisation de bonne qualité, en temps réel, pose encore des problèmes liés à la longueur du processus de convolution ou à la mobilité de l'auditeur.

La solution passe, soit par un processeur (DSP) dédicacé, soit par une simplification de la convolution.

4.1.2 Spatialisation

Le rôle de la spatialisation sonore est de restituer, par une diffusion sur haut-parleurs ou au casque, d'une part l'effet de position des sources dans l'espace, et d'autre part l'effet de salle (effet de réverbération lié à l'environnement acoustique). Cette définition mélange en effet spatialisation et auralisation.

Outre le gain en intelligibilité –particulièrement important dans le cadre de communications multi-locuteurs–, la spatialisation sonore augmente les qualités de réalisme et de naturel. Dans des scènes virtuelles, par exemple, elle renforce la sensation de présence d'objets et l'identification de l'utilisateur à l'avatar.

4.2 Le système audio de CINEMA

En pratique, le système audio développé par Nicolas Werner se présente actuellement comme suit :

1. La simulation par *salrev* est effectuée *offline* car c'est une opération très longue (des possibilités d'accélérer le processus sont actuellement un sujet de recherche dans le service de traitement du son et de l'image TSI). Le résultat de la simulation consiste en une réponse impulsionnelle de la salle.
2. Le système de reproduction est composé de plusieurs haut-parleurs, permettant de reproduire fidèlement l'ambiance acoustique dans le plan horizontal. L'ordinateur effectue, entre autres, la convolution (opération très lourde) ainsi qu'un panning d'amplitude (calcul de la répartition des signaux sur les haut-parleurs).

4.3 Communication

Afin que les différents modules de tous les partenaires du projet puissent collaborer, il nous faut concevoir un système de communication. Ce système doit être flexible et permettre de tester différents modules simultanément. Pour ce faire l'approche réseau semble la plus naturelle, elle permet de tester les programmes sur différentes machine, voire sur une même machine. Des API existent pour faciliter sa conception. De plus, nous pouvons poser plusieurs hypothèses qui nous permettent de préciser quel type d'application développer :

– les différents programmes tournant seront reliés entre-eux par un LAN. Nous pouvons donc supposer que les pertes sont minimales. Il n'y aura donc pas de routeur intermédiaire.
– Un débit minimum pourra être supposé.
– Le nombre de clients à gérer sera faible.

Ces différentes données nous poussent à penser que, d'un point de vue théorique strict, une connexion de type UDP multicast est la meilleure option. En effet, nous évitons ainsi les overheads et les délais dûs au *three-way handshake* pour chaque connexion. Chaque client peut envoyer des messages à tous les autres clients, chaque client reçoit les messages de tous les autres clients et finalement le multicasting est réalisé par le noyau et non par l'application.

Néanmoins, dans un premier temps, une simple application UDP classique devrait couvrir les besoins actuels.

Reste à définir le format (la syntaxe) des messages à envoyer : l'XML semble être le meilleur choix car, s'ils sont correctement choisis, les noms des balises seront suffisamment explicites pour être compris par chacun.

Pour rappel, l'XML (Extensible Markup Language ou langage de balisage extensible) est un standard du World Wide Web Consortium qui sert de base pour créer des langages balisés spécialisés ; c'est un « méta langage ». Il est suffisamment général pour que les langages basés sur XML, appelés aussi dialectes XML, puissent être utilisés pour décrire toutes sortes de données et de textes. Il s'agit donc partiellement d'un format de données.

Ci-dessous, un exemple partiel de messages envoyés entre deux modules :

```
<STATIC>

  <HEAD>
    <Xmin>414</Xmin>
    <Xmax>424</Xmax>
    <Ymin>165</Ymin>
    <Ymax>184</Ymax>
  </HEAD>

  <Hand side="0">
    <Xmin>186</Xmin>
    <Xmax>198</Xmax>
    <Ymin>166</Ymin>
    <Ymax>186</Ymax>
    <opening>0</opening>
  </Hand>

  <Hand side="1">
    <Xmin>500</Xmin>
    <Xmax>529</Xmax>
    <Ymin>165</Ymin>
    <Ymax>184</Ymax>
    <opening>0</opening>
  </Hand>

</STATIC>
```

Ce simple fichier XML permet de s'échanger les positions des régions d'intérêt des mains et de la tête.

Chapitre 5

Résultats

Cette section, composée principalement d'illustrations, présente les différents résultats.

Les méthode que nous avons employées tout au long de ce travail visaient principalement à permettre une utilisation temps réel de l'application sans qu'il n'y ait d'effet de saccade ou de délai. Nos caméras peuvent fonctionner à différentes résolutions : 640×480, 800×600 ou 1024×768. Nos caméras n'effectuent pas de rééchantillonage de l'image à ces différentes résolutions les portions de l'espace visibles sont donc tronquées au fur et à mesure que nous utilisons de plus petites tailles d'images.

L'utilisation d'images en 1024×768 ne nous permet pas d'obtenir un nombre suffisant d'images par seconde. Les images en 640×480 nous permettent d'obtenir un *framerate* d'environ 14 images par seconde, mais même en utilisant une optique de petite focale pour la caméra du plafond, la portion de sol visible est restreinte à tel point que l'utilisateur doit se trouver en un endroit très précis et ne peut pas en bouger ne serait-ce que de quelques centimètres. L'utilisation du mode 800×600 nous permet de conserver un taux d'images traitées par seconde acceptable (environ 11fps) tout en autorisant un déplacement de l'utilisateur ; c'est donc le mode que nous utiliserons le plus souvent.

Certaines photos d'écran présentent la première interface entièrement développée en GTK+. Celle-ci a du être partiellement réimplémentée en SDL afin de permettre une visualisation plein écran de l'image nécessaire à la réalisation d'un effet d'immersion et pour la projection sur grand écran. Pour que la sensation d'immersion de l'utilisateur soit agréable, il faut bien évidemment traiter un nombre suffisant d'images par seconde, mais aussi que la latence entre l'image acquise et l'image traitée soit faible.

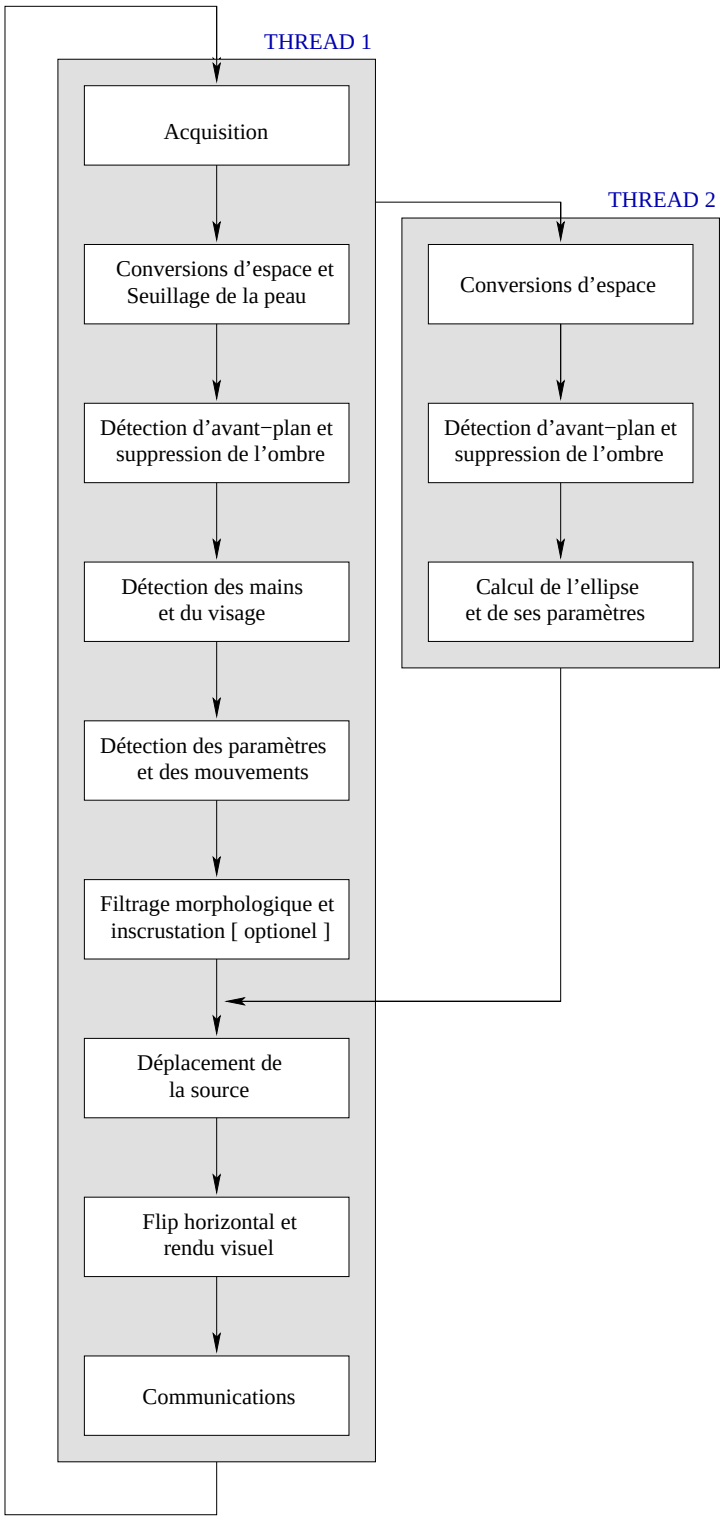

FIG. 5.1 – Le déroulement du programme. Le premier thread est en charge de la caméra frontale et est celui qui demande le plus de ressources, notamment à cause du rendu visuel. Le second thread prend en charge la caméra du plafond.

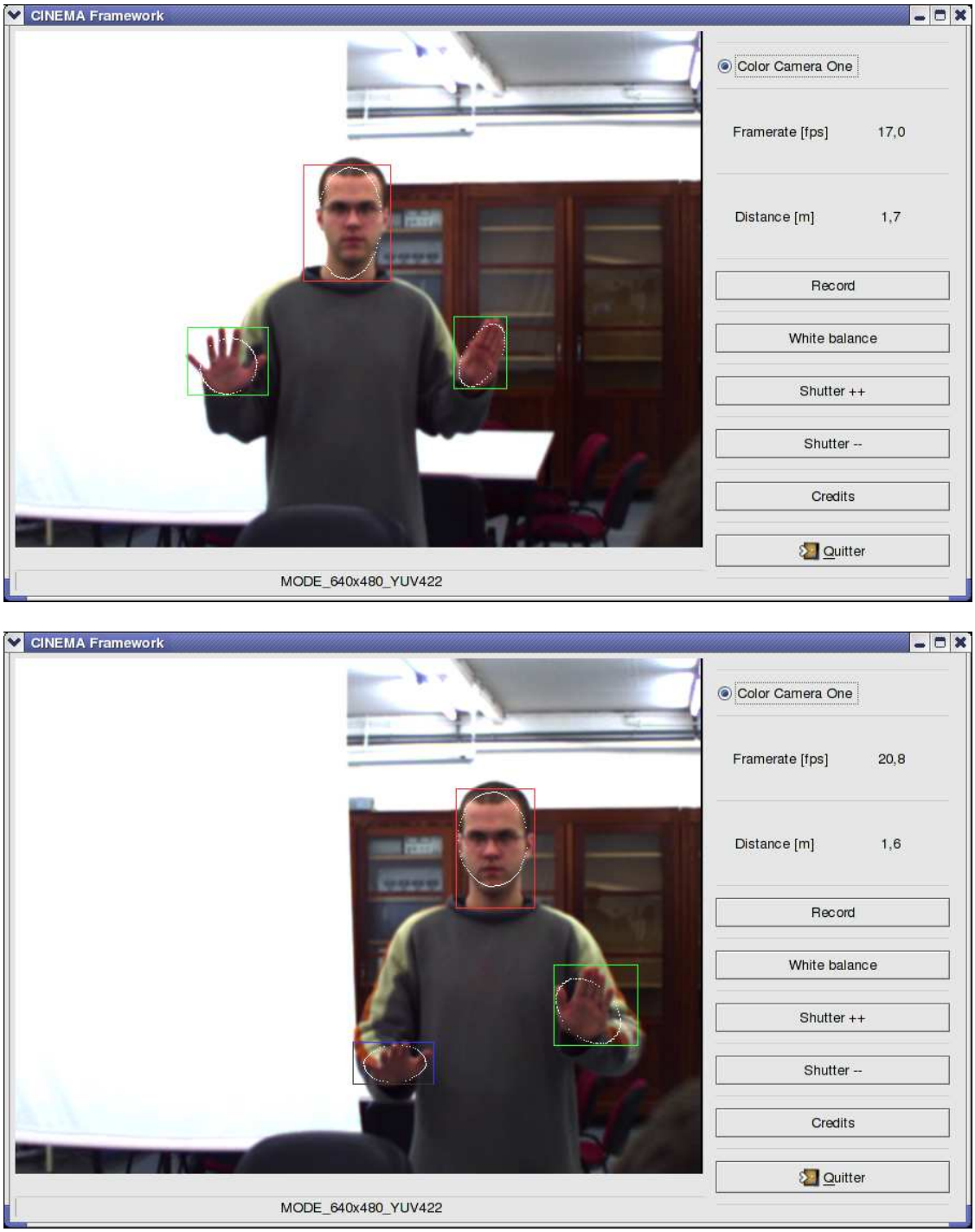

FIG. 5.2 – Le résultat illustrant la détection des mains et la reconnaissance de leur position. Le visage, une main ouverte et une main fermée sont encadrés et distingués. La distance par rapport à la caméra est mentionnée (rappelons qu'une distance de référence est toujours nécessaire).

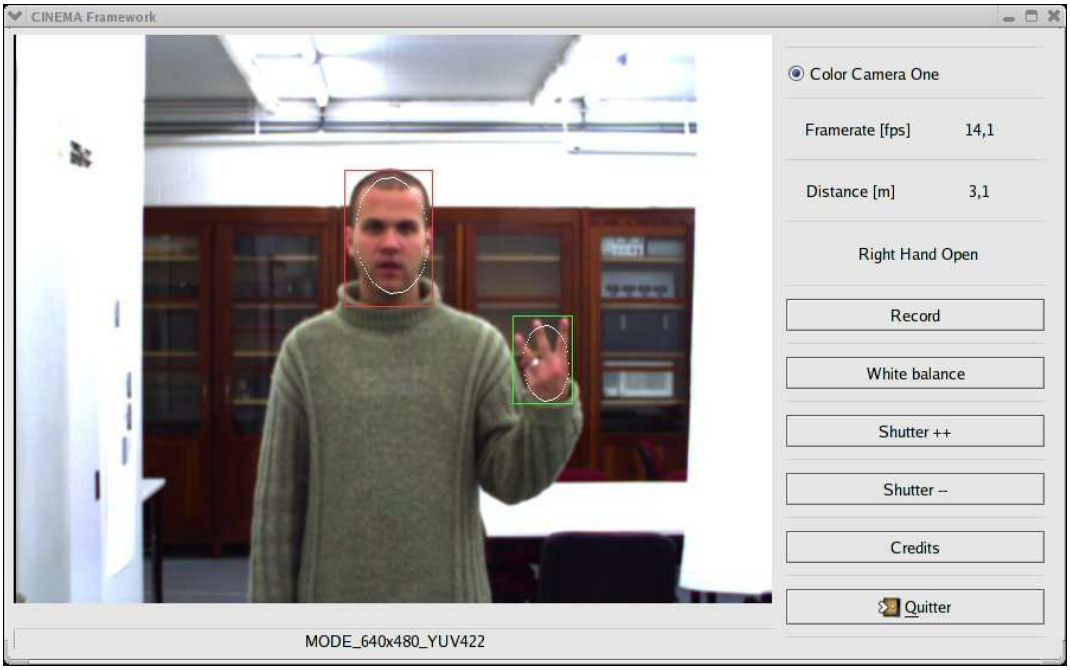

FIG. 5.3 – La détection d'une main ouverte.

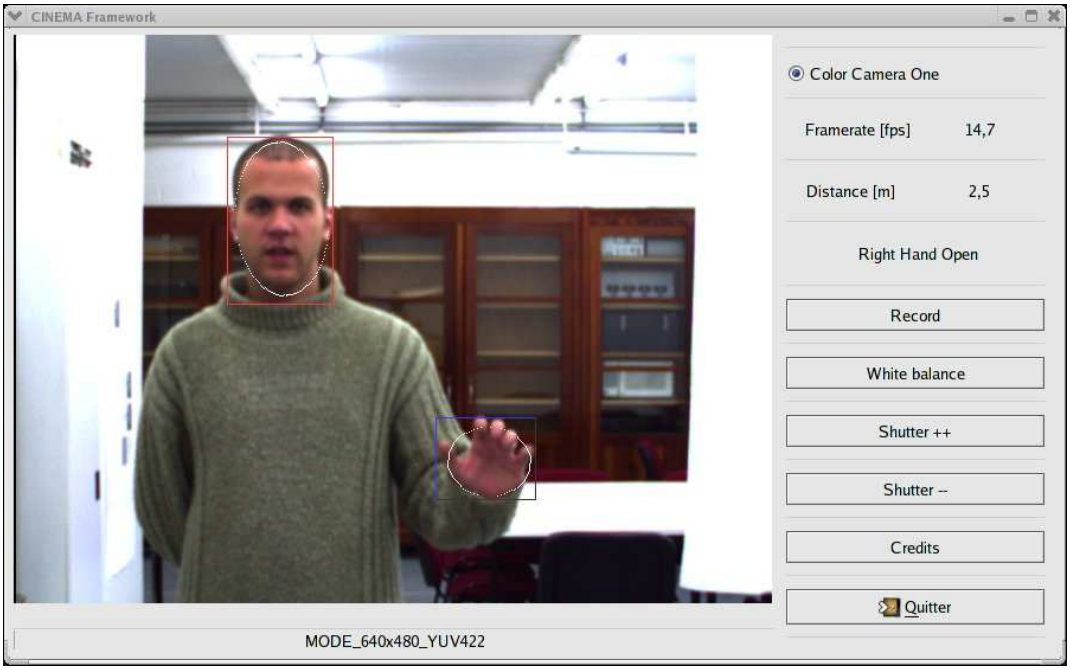

FIG. 5.4 – Le résultat illustrant la détection d'une main fermée. Remarquons que l'utilisateur porte de longues manches afin de pouvoir distinguer une main ouverte d'une main fermée.

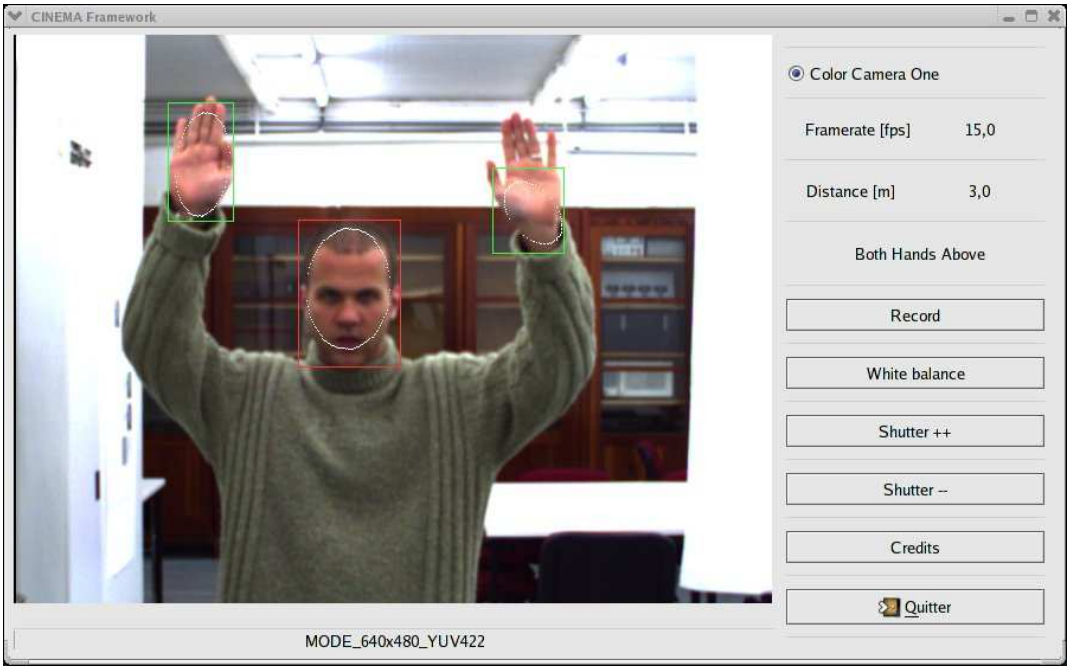

FIG. 5.5 – Le résultat final illustrant la détection de la position des mains au dessus du visage.

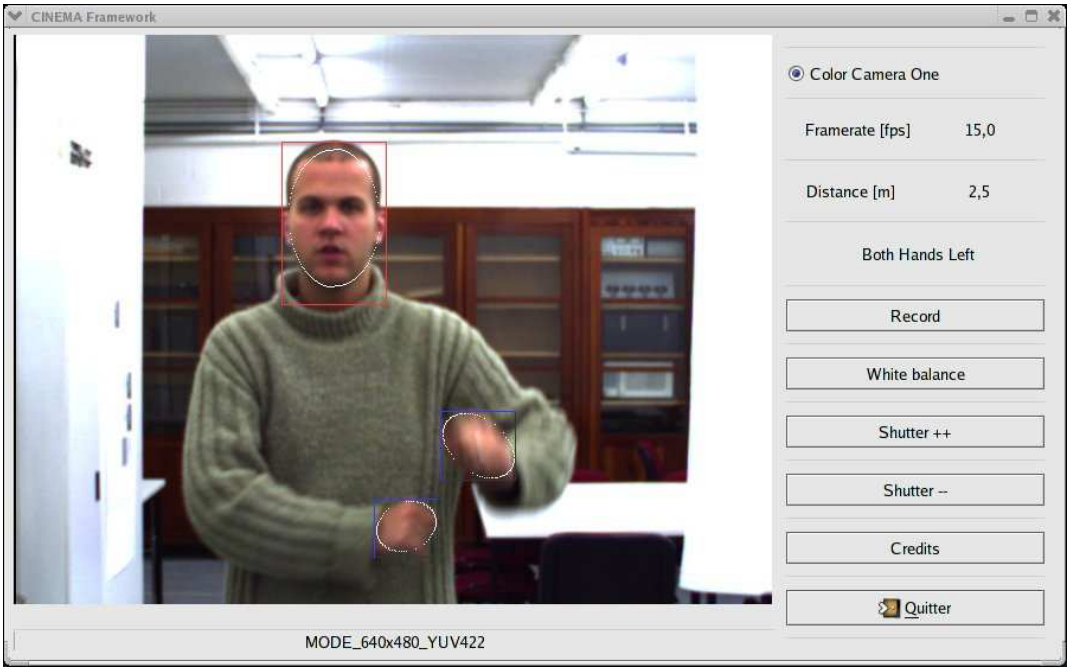

FIG. 5.6 – Le résultat illustrant la détection des deux mains d'un même côté du corps.

FIG. 5.7 – Exemples d'incrustations 2D afin que l'utilisateur soit immergé dans un décor virtuel. Une lourde étape préalable de filtrage morphologique a été appliquée afin d'éliminer le bruit et de remplir certains vides dans le corps de l'utilisateur. L'étape de flitrage ainsi que celle d'inversion gauche-droite nous font malheureusement perdre environ deux images par seconde.

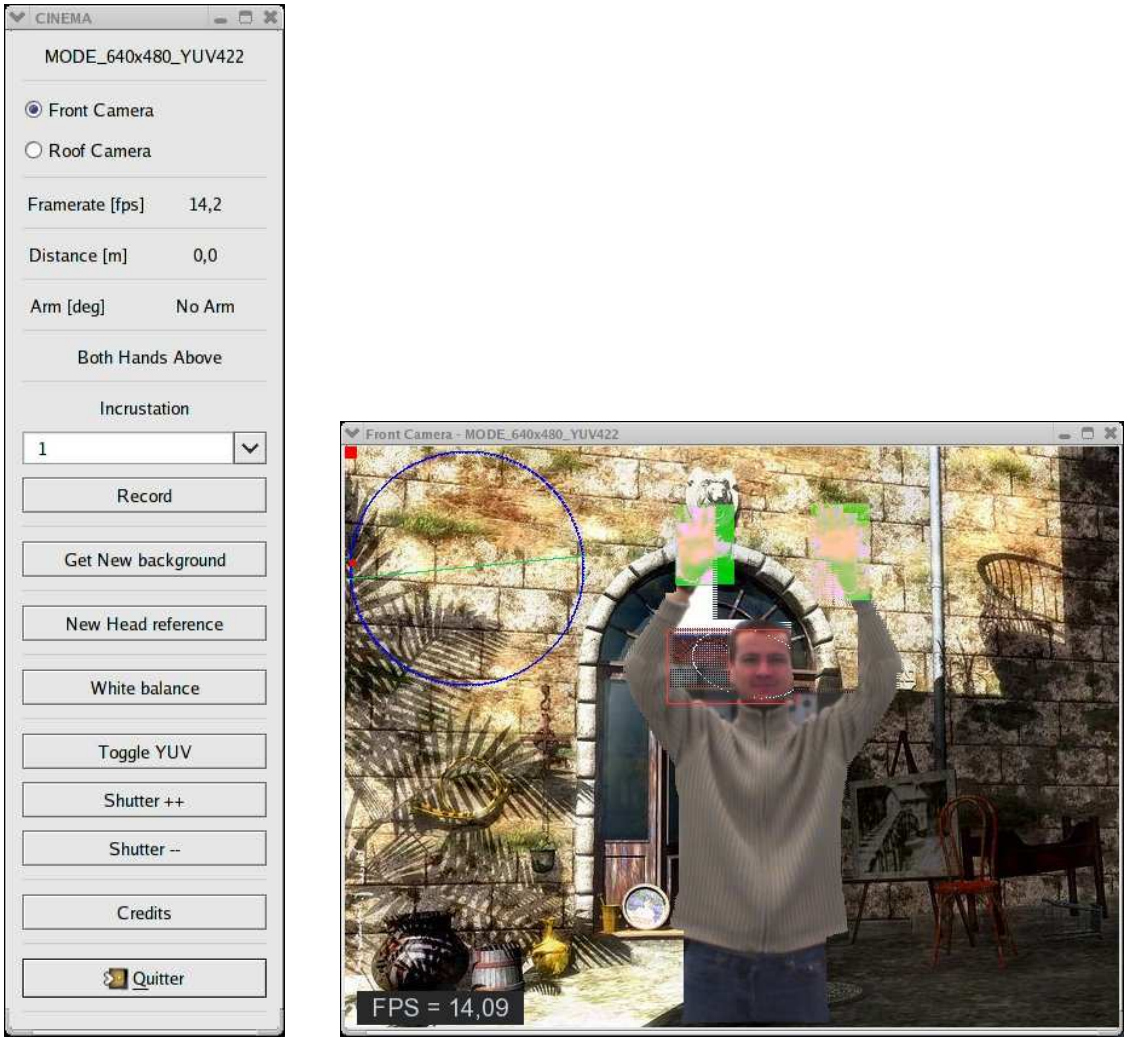

FIG. 5.8 – La nouvelle interface développée en GTK et SDL permettant d'afficher l'image en plein écran et illustrant l'affichage visuel de la reconnaissance des mains au dessus du visage. La possibilité d'afficher l'image en plein écran sur un support large participe grandement à la sensation d'immersion.

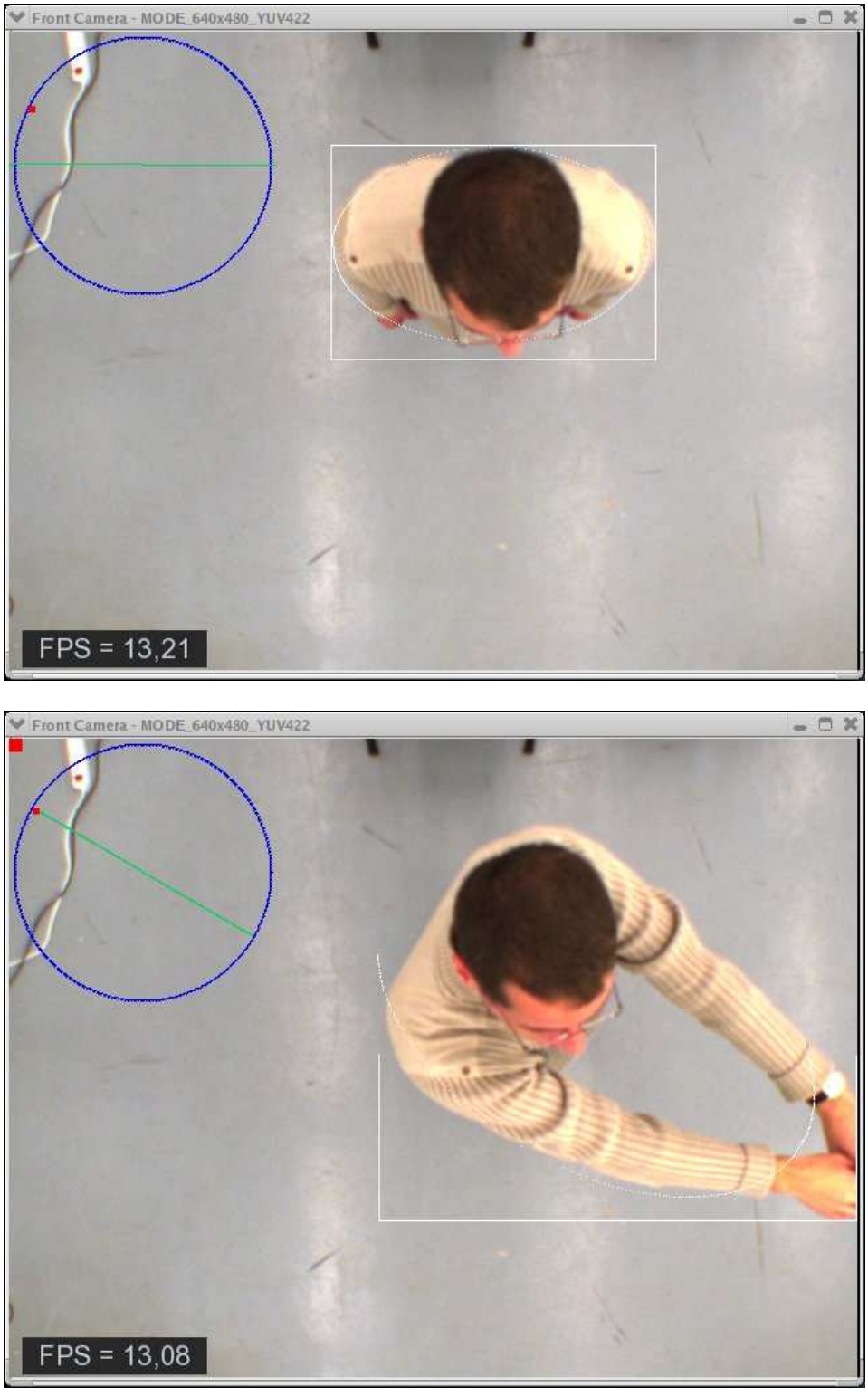

FIG. 5.9 – Vues de la caméra du plafond montrant le feedback visuel (en haut à gauche). Le carré rouge en haut à gauche montre que le bras a été détecté comme tendu. Le diamètre du cercle représente la direction des bras et le carré rouge sur le cercle représente la source qui vient d'être capturée. L'utilisateur peut ensuite en modifier la position et finalement baisser les bras pour relâcher la source.

Chapitre 6

Conclusions

De nombreux travaux en réalité virtuelle proposent l'immersion de l'utilisateur par des casques immersifs recréant un espace visuel et sonore complètement disjoint de l'environnement réel dans lequel se trouve l'utilisateur. L'interaction de l'utilisateur avec l'espace virtuel se fait par des capteurs de position de bras, mains et jambes posés à même le corps. Les moyens à mettre en œuvre sont coûteux et individualisés pour chaque utilisateur. C'est pour cette raison qu'une alternative intéressante consiste à capturer les mouvements de l'utilisateur par une ou plusieurs caméras, à segmenter l'utilisateur et lui proposer dans son champ de vision un écran sur lequel il est re-projeté mais intégré dans un décor enrichi par des éléments visuels de synthèse qui réagissent à ses actions.

Ce travail reflète les deux premières années du travail effectué dans le cadre du projet CINEMA. Nous avons commencé par montrer que l'installation du setup expérimental n'est pas une tâche à négliger et que l'installation complète de celui-ci ne se déroule pas toujours aussi vite que l'on pourrait l'espérer.

Le chapitre 2 est consacré à l'acquisition des images via nos caméras. Il est très technique, mais ce ne fut pas un ouvrage simple étant donné le peu de documentation disponible. Le chapitre 3 est consacré aux méthodes employées pour détecter certaines caractéristiques de l'utilisateur afin de pouvoir interagir avec son environnement. Nous avons finalement décrit brièvement la partie audio que nous commandons et nous avons terminé en exposant nos résultats.

Quelles conclusions pouvons nous tirer ?

Tout d'abord, il nous faut préciser que notre travail est conçu pour fonctionner en environnement contrôlé : nous avons émis un certain nombre d'hypothèses - certaines d'entre elles sont tout à fait réalistes, d'autre peuvent être plus vues comme des solutions *ad-hoc* à certaines contraintes - afin de nous faciliter la tâche ; citons par exemple la luminosité que nous supposons constante tout au long de l'expérience.

Ensuite, nous avons toujours gardé à l'esprit la contrainte de temps réel. Celle-ci nous a imposé certains choix algorithmiques pour la respecter. Les méthode employées sont donc simples mais perfectibles : la détection de peau pourrait être plus adaptative afin de mieux prendre en compte

les changements d'illumination, le méthode de détection de mouvement par suppression d'arrière plan pourrait être plus robuste au bruit, etc.

Nous sommes néanmoins parvenus à réaliser une interface homme machine simple permettant de commander une source sonore. Le peu de mouvements nécessaires pour la commander nous ont amené à effectuer une reconnaissance de mouvement élémentaire mais toutefois suffisante.

Bibliographie

[1] *IIDC 1394-based Digital Camera Specification*.

[2] `http://asi.insa-rouen.fr/~lfallet/docs/sem1/FireWire.pdf`. Document disponible sur Internet.

[3] `http://freshmeat.net/projects/coriander/`. Document disponible sur Internet.

[4] `http://www.alliedvisiontec.com/`. Document disponible sur Internet.

[5] `http://www.arnaudfrichphoto.com`. Document disponible sur Internet.

[6] `http://www.commentcamarche.net/pc/firewire.php3`. Document disponible sur Internet.

[7] `http://www.linux1394.org`. Document disponible sur Internet.

[8] `http://www.videredesign.com/support_linux_kernel.htm`. Document disponible sur Internet.

[9] `http://sourceforge.net/projects/libdc1394/`. Document disponible sur Internet.

[10] `http://sourceforge.net/projects/libraw1394`. Document disponible sur Internet.

[11] `http://www.tele.ucl.ac.be/PEOPLE/DOUXCHAMPS/ieee1394/cameras/`. Document disponible sur Internet.

[12] Mohan Trivedi Andrea Prati, Ivana Mikic and Rita Cucchiara. Detecting moving shadows : algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7), July 2003.

[13] J. Bjørnstrup. Estimation of human body segment parameters : Historical background. Technical report, Laboratory of Image Analysis, Institute of Electronic Systems, Aalborg University, available at `http://www.cvmt.auc.dk/~jorgen/PhD/EHBSP_background/`, October 1995.

[14] C.-C. Chiang, W.-K. Tai, M.-T. Yang, Y.-T. Huang, and C.-J. Huang. A novel method for detecting lips, eyes and faces in real time. *Real-Time Imaging*, 9(4) :277–287, 2003.

[15] R. Dardenne and M. Van Droogenbroeck. Acquisition vidéo au moyen d'une caméra IEEE 1394. *Linux Magazine France*, (69) :54–59, Février 2005.

[16] R. Drillis and R. Contini. Body segment parameters. Technical Report Report no. 1166-03, New York, New York University School of Engineering Science., 1966. (Office of Vocational Rehabilitation, DHEW).

[17] S. Malassiotis F. Tsalakanidou and M. Strintzis. Face localization and authentication using color and depth images. *IEEE Transactions on Image Processing*, 14(2) :152–168, February 2005.

[18] D. A. Forsyth and J. Ponce. *Computer Vision : a Modern Approach*. Prentice Hall, 2003.

[19] R. Green and L. Guan. Quantifying and recognizing human movement patterns from monocular video images - part 1 : A new framework for modeling human motion. *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Image and Video-Based Biometrics*, November 2003.

[20] M. Harville. Stereo person tracking with adaptive plan-view templates of height and occupancy statistics. *Image and Vision Computing*, 22(22) :127–142, 2004.

[21] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *IEEE Frame-Rate Workshop*, 1999.

[22] R.-L. Hsu, M. Abdel-Mottaleb, and A. Jain. Face detection in color images. *IEEE Trans. PAMI*, 24(5) :696–706, May 2002.

[23] P. Jonker, J. Caarls, and W. Bokhove. Fast and accurate robot vision for vision based motion. In Springer-Verlag, editor, *Proc. 4th Int. Workshop on Robocup*, pages 149–158, Melbourne, Australia, 2001.

[24] M. Kikuchi and K. Fukushima. Pattern recognition with eye movement : A neural network model. *International Joint Conference on Neural Networks*, 2(2), 2000.

[25] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Trans. PAMI*, 23(4) :349–361, April 2001.

[26] M. Nixon and J. Carter. In *Proceedings of the sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 139–144, May 2004.

[27] S. Persa and P. Jonker. Hybrid tracking system for outdoor augmented reality. In *Proc. 2nd Int. Symposium on Mobile Multimedia Systems and Applications MMSA2000*, 2000.

[28] S. Persa and P. Jonker. On positioning for augmented reality systems. In *IEEE Benelux Signal Processing Symposium*, Hilvarenbeek, The Netherlands, March 2000.

[29] S. Persa and P. Jonker. Real-time computer vision system for mobile robot. In *Intelligent Robots and Computer Vision XX : Algorithms, Techniques, and Active VisionNN*, 2000.

[30] E. Polat, M. Yeasin, and R. Sharma. A 2d/3d model-based object tracking framework. *Pattern Recognition*, 36(9) :2127–2141, September 2003.

[31] E. Polat, M. Yeasin, and R. Sharma. Robust tracking of human body parts for collaborative human computer interaction. *Computer Vision and Image Understanding*, 89(1) :44–69, Janvier 2003.

[32] M. Piccardi R. Cucchiara, C. Grana and A. Prati. Detecting objects, shadows and ghosts in video streams by exploiting color and motion information. In *11th International Conference on Image Analysis and Processing, ICIAP 2001*, pages 360–365, 2001.

[33] M. Reuvers, R. Kleihorst, H. Broers, and B. Kröse. A smart camera for face recognition. In *Fourth IEEE Signal Processing Symposium*, pages 155–158, Hilvarenbeek, The Netherlands, April 2004.

[34] E. Sahin and P. Gaudiano. Visual looming as a range sensor for mobile robots. In *Fifth International Conference on Simulation of Adaptive Behavior*, Zurich, Switzerland, 1998.

[35] G. Salgian and D. Ballard. Visual routines for vehicle control. In *International Conference on Computer Vision*, 1998.

[36] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Tom Robbins, 2001.

[37] J. Terrillon and S. Akamatsu. Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images. pages 54–61, 2000.

[38] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[39] Z. Zeng and S. Ma. An efficient vision system for multiple car tracking. In IEEE, editor, *16th International Conference on Pattern Recognition, Volume 2*, pages 609–612, 2002.

[40] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition : A literature survey. *ACM Comput. Surv.*, 35(4) :399–458, 2003.

Je remercie mon promoteur *Marc Van Droogenbroeck* ainsi que les membres du jury : messieurs *Jacques Destiné*, *Jean-Jacques Embrechts* et *Pierre Leclercq*. Je remercie également, parmi les personnes de Montefiore, celles et ceux avec qui je passe le plus clair de mon temps. Ils se reconnaitront :)

Merci.