Téléprésence, immersion et interactions pour le reconstruction 3D temps-réel

Benjamin Petit

▶ **To cite this version:**

Benjamin Petit. Téléprésence, immersion et interactions pour le reconstruction 3D temps-réel. Mathématiques générales [math.GM]. Université de Grenoble, 2011. Français. NNT : 2011GRENM007 . tel-00584001

HAL Id: tel-00584001

https://theses.hal.science/tel-00584001

Submitted on 7 Apr 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques et Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Benjamin PETIT

Thèse dirigée par **Edmond BOYER** et
codirigée par **Bruno RAFFIN**

préparée au sein des laboratoires **LJK et LIG**
dans l'école doctorale **MSTII**

Téléprésence, immersion et interaction pour la reconstruction 3D temps-réel

Thèse soutenue publiquement le **21 Février 2011**
devant le jury composé de :

Mme. Indira, THOUVENIN
Enseignant chercheur à l'Université de Technologie de Compiègne, Président
Mr. Bruno, ARNALDI
Professeur à l'INSA Rennes, Rapporteur
Mme. Saida, BOUAKAZ
Professeur à l'Université Claude Bernard Lyon 1, Rapporteur
Mr. Edmond, BOYER
Directeur de recherche à l'INRIA Grenoble, Membre
Mr. Bruno, RAFFIN
Chargé de recherche à l'INRIA Grenoble, Membre

Remerciements

Je voudrais commencer par remercier Edmond et Bruno pour avoir encadrer ma thèse. Ce fut un plaisir de travailler avec vous.

Merci également aux membres de mon jury d'avoir accepté de rapporter cette thèse. Merci pour vos commentaires très constructifs.

Pendant ma thèse j'ai eu l'occasion de travailler avec différentes personnes. Ces collaborations ont été très enrichissantes. Je voudrais remercier plus spécifiquement Jean-Denis qui m'a aidé à remettre sur pied la plateforme Grimage, Thomas avec qui j'ai passé de longues heures à développer les applications et démonstrations de ma thèse et enfin Hervé pour son excellent support sur la plateforme Grimage. J'aimerais remercier également Clément et Florian pour m'avoir transmis leur savoir sur la plateforme Grimage, Nicolas et Jean-François pour leur aide technique.

Merci à mes deux équipes : PERCEPTION et MOAIS pour leur accueil respectif. Plus particulièrement à Antoine pour son implication dans les parties théoriques de ma thèse et à tous les autres pour leur implication dans la vie de l'équipe au labo et en dehors du labo...

Merci aux partenaires de l'ANR Dalia et plus particulièrement à Jean-Sébastien, Benoît et Joeffrey avec qui nous avons passé quelques journées à développer à distance notre application de téléprésence.

Enfin je finirais ces remerciements par tout ceux qui m'ont soutenu (supporté ?) pendant ma thèse. Mon nouveau cercle d'amis Grenoblois, mes anciens copains Lyonnais, toute ma famille et plus particulièrement Julie qui m'a suivi à Grenoble et encouragé pendant ces 3 années.

Table des matières

Introduction

« Cyberspace. A consensual hallucination experienced daily by billions of legitimate operators, in every nation, by children being taught mathematical concepts... A graphic representation of data abstracted from banks of every computer in the human system. Unthinkable complexity. Lines of light ranged in the nonspace of the mind, clusters and constellations of data. Like city lights, receding. »
William Gibson

Chapitre 1

Introduction

Dans les années 1980, William Gibson, auteur de science-fiction, a inventé le terme de *Cyberspace* dans son roman *Neuromancer* [23] (*Le Neuromancien* en français) pour désigner un espace d'information virtuel, qui pourrait aujourd'hui être apparenté à internet. Dans *Snow Crash* de Neal Stephenson [73], écrit en 1992, ce terme est étendu par l'auteur, qui crée le concept de *Metaverse* pour décrire la vision qu'il a d'un univers virtuel en 3D complètement immersif. Dans son roman, les utilisateurs du *Metaverse* accèdent à l'espace virtuel au travers d'un terminal. Ceux-ci sont plus ou moins évolués et contraignent l'apparence virtuelle des utilisateurs ainsi que leur perception de l'univers virtuel. L'utilisation d'un terminal publique de mauvaise qualité a pour conséquence l'incarnation dans un avatar en niveau de gris et la perception du monde à travers des lunettes en faible résolution. Neal Stephenson popularisa ainsi le terme *avatar*, dérivé de *avatăra*, qui en Sanskrit désigne l'incarnation d'un dieu sur terre.

Depuis, l'imaginaire collectif a été alimenté par des productions cinématographiques telles que *Matrix* de Andy et Larry Wachowski ou *Avalon* de Mamoru Oshii, films qui décrivent des environnements virtuels en 3D où l'humain est complètement immergé dans un univers plus vrai que nature.

Poussé par ce fantasme de réalisme virtuel, les progrès en informatique ont permis l'implémentation d'univers artificiels de plus en plus réalistes, peuplés d'avatars de bonne qualité qui ont des possibilités d'interaction de plus en plus avancées et permettent aux utilisateurs d'avoir tous leurs sens impliqués.

1.1 Contexte

Les mondes virtuels persistants, aussi couramment appelés *MMO* ou *MMOG* (pour *Massively Multiplayer Online Game*) sont en pleine expansion. D'abord exclusivement réservés aux jeux multi-joueurs en ligne, et plus spécifiquement aux

FIGURE 1.1 – Vue artistique de l'avatar de *Hiro*, personnage principal de *Snow Crash*, immergé dans le *Metaverse*.

jeux de rôle, ces univers se sont étendus aux réseaux sociaux avec, entre autre, l'apparition de *Second Life*®.

Un monde virtuel est un univers créé artificiellement dans lequel une communauté d'utilisateurs distants s'immergent dans le but de communiquer et d'interagir. Pour cela, les utilisateurs doivent :

- avoir l'impression d'être présent : c'est la sensation d'immersion,
- donner l'impression d'être présent : c'est la présence visuelle véhiculée par l'avatar,
- pouvoir agir sur l'environnement virtuel : c'est la présence mécanique, les actions que l'on peut faire exécuter à son avatar.

Il existe aujourd'hui de multiples techniques et périphériques associés, qui permettent, au moins partiellement, de répondre à ces trois contraintes.

L'immersion peut-être réalisée grâce à un périphérique de visualisation. Ces derniers peuvent aller du simple écran d'ordinateur, aux solutions plus complexes comme les environnements de type *CAVE* qui permettent une projection du monde virtuel tout autour de la personne. La technologie actuelle permet également l'affichage en relief (dit 3D stéréoscopique) qui apporte encore plus de réalisme aux scènes visualisées.

La qualité des avatars a également fortement évolué au cours des dernières années, à l'origine un simple nom ou *pseudo* pouvait nous représenter dans les espaces collaboratifs, il a ensuite été accompagné d'une imagette ou photo nous

personnifiant encore plus. Aujourd'hui, on peut se créer un avatar 3D fidèle à son image. Les périphériques vidéos, comme la webcam, qui accompagne maintenant la plupart des ordinateurs personnels, offrent même la possibilité d'apparaître aux autres directement, comme dans les solutions de téléconférence telles que *Skype*™.

Le contrôle de nos avatars lui aussi profite des dernières avancées technologiques. Bien que largement contraint par l'utilisation du clavier et de la souris, le contrôle des actions pré-programmées de notre représentation virtuelle prend un nouveau départ grâce aux nouveaux périphériques accessibles au grand public. La *Wiimote*® de *Nintendo*® ou le projet *Kinect*™ (ex *Natal*) de *Microsoft*® offrent de nouvelles possibilités en matière de contrôle et d'actions réalisables par nos avatars.

Mais n'est-ce pas limitant de n'être représenté que par une image en 2D ou par un modèle synthétique 3D qui a du mal à véhiculer nos émotions ? N'est-il pas frustrant de devoir utiliser un contrôleur pour effectuer les quelques actions pré-programmées de notre avatar ?

Le domaine de la vision par ordinateur, ou vision artificielle, tente de reproduire la vision humaine afin de permettre aux machines équipées d'une ou plusieurs caméras d'avoir une perception visuelle semblable à celle de l'humain. Des systèmes de vision, qui comprennent un réseau de caméras et un cluster de machines ont vu le jour. Ceux-ci apportent une réponse partielle à ces problèmes, car ils ont maintenant les capacités de modéliser les formes et les mouvements en temps-réel et en 3D.

1.2 Problématiques

C'est dans ce contexte que j'ai placé mes travaux de thèse. J'ai essayé d'apporter des réponses et des solutions à l'utilisation des systèmes de vision multi-caméra comme périphériques d'entrée dans les mondes virtuels.

Il a été montré qu'il est possible d'utiliser un réseau de caméras relié à un cluster de PC pour modéliser en 3D toute personne présente dans un espace d'acquisition, et cela en temps-réel [1]. Ces travaux ont permis d'utiliser ces informations sur l'utilisateur pour des applications de réalité virtuelle [49]. Le modèle 3D produit ainsi que les textures issues des images prises par les caméras permettent d'avoir une représentation fidèle de l'utilisateur, utile pour se sentir immergé dans un espace virtuel et pour donner cette impression à d'autres. Basées sur une détection de collision entre le modèle produit et celui d'objets virtuels, des actions sont également réalisables [3].

Sur la base de ces travaux, j'ai étudié les possibilités d'immerger encore plus l'utilisateur afin d'augmenter son sentiment de présence dans le monde virtuel. J'ai, pour cela, exploré le couplage des systèmes multi-caméra avec des périphériques de visualisation plus immersifs.

J'ai également essayé d'améliorer ses possibilités d'interaction. Pour cela, il a fallu que j'explore le potentiel des informations récupérables à partir du modèle 3D de la personne et des images prises par les caméras pour imaginer de nouvelles métaphores d'interaction.

Enfin, la technologie accessible ne permettait pas à l'utilisateur d'être présent à distance (téléprésence) sous la forme d'un avatar le représentant fidèlement et avec des possibilités d'interaction avancées. Il y avait encore des difficultés à surmonter comme les débits nécessaires au transport de l'information 3D des utilisateurs ou les faibles latences requises par le besoin d'interactivité du système.

1.3 Contributions

Mes contributions dans le domaine de l'utilisation de réseaux de caméras dans les applications de réalité virtuelle se sont orientées selon trois axes présentés dans la suite. Les deux premiers axes sont des contributions plus expérimentales alors que le troisième axe est une contribution plus méthodologique.

1.3.1 Téléprésence et collaboration à distance

Mes travaux de thèse sur la téléprésence et la collaboration à distance ont été réalisés autour du projet *ANR Dalia* mené conjointement avec les équipes de l'*INRIA Bordeaux* et du *LIFO d'Orléans*.

Nous avons montré que les systèmes multi-caméra peuvent être utilisés pour des applications de téléprésence et de collaboration à distance. Pour cela nous avons développé une application qui implique plusieurs plateformes de reconstruction 3D situées à plusieurs centaines de kilomètres les unes des autres et permet à des utilisateurs distants de se retrouver dans un monde virtuel commun et d'interagir ensemble.

Nous avons réaliser l'adaptation des briques d'application existantes pour qu'elle passe à l'échelle d'une application distribuée sur une grille de calcul. Les difficultés à surmonter sont liées aux contraintes d'interactivité de tout système de collaboration à distance, comme le transfert sur le réseau de l'apparence physique

et mécanique des utilisateurs à un taux de rafraîchissement assez grand et à une latence assez faible pour leur permettre d'être réactifs et d'interagir ensemble.

Une première plateforme de démonstration a été présentée lors de la conférence VRST'08 à Bordeaux [58]. Ce démonstrateur se composait de deux plateformes mobiles de reconstruction 3D temps-réel reliées entre elles. Les utilisateurs pouvaient y mettre leurs mains qui étaient aussitôt reconstruites en 3D et immergées dans un espace 3D commun. Là, ils pouvaient, en collaboration, jouer avec un modèle 3D déformable animé par un logiciel de simulation physique.

Par la suite, une publication scientifique a été présentée sur le sujet lors de la conférence *3DTV-Con'09* [57] à Potsdam, en Allemagne, ainsi que dans un article pour l'*International Journal of Digital Media Broadcasting* (*IJDMB*) dans une série spéciale : *3DTV Theory and Practice* [59].

Nous avons finalement réalisé un second démonstrateur qui permet à plusieurs utilisateurs présents à Bordeaux, Orléans et Grenoble, de se retrouver dans un espace virtuel commun et d'interagir ensemble, en 3D, en utilisant les moyens techniques de chacun des sites. Ce démonstrateur utilise le réseau *Grid'5000*, réseau qui permet des débits bien supérieurs aux débits possibles avec l'internet standard et de faibles latences. Une vidéo a été produite sur la base de ce démonstrateur et publiée dans le cadre de la conférence *ACM Multimedia'10* [56].

1.3.2 Immersion et interaction à la première personne

Afin d'immerger l'utilisateur dans un espace virtuel et de renforcer son sentiment de présence et d'identification à l'avatar qu'il contrôle, nous avons couplé un système multi-caméra permettant de modéliser en 3D l'utilisateur, avec un dispositif de visualisation à la première personne. Nous avons utilisé pour cela un casque de réalité virtuelle (*Head Mounted Display* ou *HMD*) et un périphérique qui permet de suivre la tête de l'utilisateur afin de lui procurer une vision unidirectionnelle, à la première personne, de la scène virtuelle.

Cette immersion visuelle, couplée à la modélisation 3D temps-réel, permettent à l'utilisateur de voir ses mains modélisées en 3D et d'interagir de façon co-localisée avec des objets virtuels. Cette intégration pose le problème de l'alignement des repères et de la synchronisation des données issues des différents périphériques.

Nous avons également intégré une caméra portée sur la tête de l'utilisateur pour améliorer la qualité visuelle et mécanique du modèle virtuel de l'utilisateur en apportant un point de vue plus précis sur la partie de son corps qu'il est en train d'observer.

À cette occasion, nous avons créé le projet *VGate* (pour *Virtualization Gate*). L'idée était de créer un portail de virtualisation, une interface qui permet à un utilisateur d'être virtualisé et immergé dans un monde virtuel en mixant les moyens de notre système de vision et les possibilités offertes par les périphériques modernes de visualisation.

Ce projet a été valorisé par la création d'une vidéo de présentation du concept, vidéo soumise pour la conférence *Siggraph'09* à la Nouvelle-Orléans, États-Unis, et concrétisée par une démonstration dans la partie *Emerging Technologies* de la conférence [60]. Les utilisateurs pouvaient profiter d'une reproduction de notre plateforme à échelle humaine où ils étaient modélisés en 3D en temps-réel et immergés dans un monde virtuel par le biais d'un casque de réalité virtuelle. Ils pouvaient alors interagir, à la première personne, avec des objets virtuels.

1.3.3 Champs de vitesse du modèle 3D

Dans le but d'obtenir plus d'informations sur le modèle 3D produit par le système de vision multi-caméra, comme la vitesse, l'accélération ou les forces, j'ai développé une méthode pour calculer un champs de vitesse 3D lié à la surface du modèle 3D de l'utilisateur. Cette méthode repose sur l'observation des mouvements dans les images prises par les caméras, basées sur des techniques de flux optique et de détection et mise en correspondance de points d'intérêts 2D et 3D. Je recherche le déplacement instantané optimal du modèle 3D en fonction de ces observations et d'un modèle de déformation de la surface. Ces informations de déplacement sont ensuite converties en champs de vitesse instantanée du modèle. Ce champs de vitesse peut ainsi être utilisé pour enrichir des calculs de collisions en permettant de calculer des accélérations qui seront utiles pour moduler la force de collision et déclencher des interactions plus réalistes.

1.3.4 Valorisations

Outre les publications [57, 59, 56] et démonstrations [58, 60] listées précédemment, j'ai participé activement au développement de la plateforme *GrImage* et au projet de développement logiciel *FlowVR*.

1.3.4.1 La plateforme GrImage

La plateforme *GrImage* est un système de vision multi-caméra pour la réalité virtuelle. La majorité de mes travaux de thèse a été réalisé pour créer ou améliorer

les applications qui utilisent la plateforme *GrImage*. En parallèle de mes travaux de recherche, j'ai joué un rôle important dans la maintenance de la partie technique de cette plateforme. J'ai contribué à la remise en place des démonstrateurs créés par mes prédécesseurs et au développement matériel et logiciel de nouvelles solutions.

Pendant ma thèse, j'ai participé à plusieurs démonstrations impliquant la version portable de la plateforme *GrImage* [3], telles que les *40 ans de l'INRIA* à Lille en Décembre 2007 et la *Fête de la Science* au Grand Palais de Paris en Octobre 2008.

1.3.4.2 FlowVR

FlowVR est un intergiciel dédié au développement d'applications interactives pour les architectures distribuées. C'est un logiciel libre développé par l'équipe *MOAIS* depuis 2003. La plateforme *GrImage* est extrêmement liée au développement de ce logiciel car il est la colonne vertébrale de toutes les applications interactives développées sur la plateforme.

J'ai donc été amené à participer au projet de développement de *FlowVR*, d'intégrer ses nouvelles fonctionnalités dans les applications de la plateforme *GrImage* et de proposer des extensions et des corrections à l'équipe de développement.

1.4 Organisation

Dans le chapitre §2, après un état de l'art des environnements virtuels et des possibilités d'immersion et d'interaction existantes, dans lequel les systèmes de vision et leurs utilisations comme périphérique d'entrée en réalité virtuelle seront présentés (section §2.1), la section §2.2 fera une synthèse de la conception d'un système multi-caméra en prenant le contexte de la plateforme *GrImage*.

Le chapitre §3 présentera la création d'une application de téléprésence et de collaboration à distance basée sur un tel système multi-caméra.

Dans le chapitre §4, la solution de couplage de notre plateforme multi-caméra avec un système de visualisation avancé permettant des interactions à la première personne. Puis dans le chapitre §5, une solution sera proposée pour améliorer la qualité visuelle et mécanique des modèles issues du processus de reconstruction 3D de la plateforme multi-caméra.

Enfin, une technique qui permet de calculer des champs de vitesse dense sur la surface du modèle 3D et son utilisation possible pour enrichir les interactions sera présentée dans le chapitre §6.

Le chapitre §7 conclura sur un bilan de mes travaux et des perspectives futures.

I

Systèmes de vision et téléprésence

« If, as it is said to be not unlikely in the near future, the principle of sight is applied to the telephone as well as that of sound, earth will be in truth a paradise, and distance will lose its enchantment by being abolished altogether. »
Arthur Strand, 1898

Chapitre 2

Systèmes de vision et réalité virtuelle

2.1 État de l'art

2.1.1 Environnements virtuels

2.1.1.1 Mondes virtuels et avatars

Un monde virtuel est un monde créé artificiellement. Il rassemble une communauté d'utilisateurs, représentés par leurs avatars, qui peuvent évoluer dans le monde, se déplacer, se rencontrer et effectuer des actions en collaboration ou non. Certains reproduisent la réalité en copiant les lois de la physique quand d'autres sont complètement fantastiques.

On distingue deux genres de mondes virtuels :

- Les mondes qui constituent la base d'un jeu en ligne, jeu dit massivement multi-joueurs (MMO ou MMOG), comme *Everquest*® [1] ou *World of Warcraft*® [2].
- Les simulateurs de monde. Ces derniers ne sont pas considérés comme des jeux car ils ne comportent pas de but ou de règles. On peut citer ici *Le Deuxième Monde*® [3], monde virtuel francophone disparu en 2001, ou *Second Life*® [4], qui rencontre un succès croissant ces dernières années. Véritable lieu de rencontre, il est pris d'assaut par les professionnels et les universitaires qui n'hésitent pas à organiser des séminaires, des campagnes publicitaires ou encore des recrutements au sein de leurs espaces virtuels. Par exemple, la réunion du comité de programme de *IEEE VR* se déroule depuis 2009 sur *Second Life*®.

1. http://www.everquest.com
2. http://www.worldofwarcraft.com
3. http://fr.wikipedia.org/wiki/Le_Deuxi%C3%A8me_Monde
4. http://secondlife.com

Notre présence dans ces mondes virtuels est assurée par le biais de notre avatar. Cet avatar est le plus souvent un modèle 3D synthétique, créé à partir d'une bibliothèque de caractéristiques physiques. Ils peuvent nous ressembler plus ou moins fidèlement suivant les possibilités du logiciel utilisé et le temps passé à leur création (voir figure 2.1).

FIGURE 2.1 – Exemples d'avatars dans *Second Life*, créés pour ressembler aux joueurs qui les contrôlent.

Ces mondes virtuels sont très intéressants du point de vue de la recherche. Ils intéressent non seulement les chercheurs en technologie de l'information, qui essayent d'améliorer le sentiment de présence [64] au sein de ces mondes, en jouant sur le réalisme du graphisme, les périphériques d'immersion, ou encore le rendu des émotions, mais également les chercheurs en sciences humaines et sociales qui voient là un terrain d'expérimentation grandeur nature où il est plus aisé de récupérer des informations statistiques sur des comportements sociaux [5].

2.1.1.2 L'immersion dans les mondes virtuels

Afin de s'immerger dans ces mondes virtuels, le grand public est encore bien souvent limité à l'utilisation d'un moniteur ou d'une télévision et d'un système audio 2.1. Depuis 2009, les téléviseurs ou moniteurs en relief sont disponibles dans le commerce. Ils sont bien souvent improprement appelés téléviseurs 3D alors qu'ils ne fournissent pas réellement une image en 3 dimensions, mais juste le relief dans une direction. Ils permettent à l'utilisateur de percevoir une image en relief grâce aux techniques de stéréoscopie. Cette sensation renforce le sentiment de présence de l'utilisateur dans les mondes virtuels où il évolue. Si on l'associe avec un système de son spatialisé, l'expérience devient encore plus immersive.

Encore peu accessibles à cause de leur prix, les casques ou lunettes de réalité virtuelle, appelés *HMD* pour *Head Mounted Display* en anglais, existent depuis longtemps sur les marchés spécialisés, principalement pour la simulation militaire

(voir figure 2.2). Ils sont composés d'une paire d'écrans placés devant les yeux de l'utilisateur et diffusant des images stéréoscopiques. Associés à un dispositif de suivi de la position de la tête de l'utilisateur, ils renforcent le sentiment de présence de l'utilisateur en lui procurant une vision en relief, à la première personne, du monde virtuel. Le degré d'immersion dépend fortement des spécificités techniques du casque utilisé ; Plus la résolution des écrans et le champs de vision couvert sont grands, plus le sentiment de présence sera fort. Un état de l'art détaillé des *HMDs* sera présenté dans la section §4.2.1.2.

FIGURE 2.2 – Exemples de *HMD* : le *eMagin Z800* (gauche) et le *Virtual Eye* (droite).

Un autre type de périphérique d'affichage permet une bonne immersion, il s'agit des environnements immersifs basés sur des vidéo-projecteurs. Le plus répandu, et le plus immersif, est le *CAVE* [14], un environnement cubique à l'intérieur duquel une ou plusieurs personnes peuvent se tenir et dont chaque face sert d'écran (voir figure 2.3). Les premiers *CAVEs* utilisaient 4 projecteurs (pour les faces de face, gauche, droite et le sol), les plus récents utilisent les 6 faces du cube, voire des surfaces hexagonale utilisant 15 projecteurs et permettent d'afficher des images en stéréoscopie. Un état de l'art détaillé des *CAVEs* sera présenté dans la section §4.2.1.1.

FIGURE 2.3 – Principe du *CAVE* (gauche) et exemple d'application (droite).

L'avantage des environnements de type *CAVE* face à ceux basés sur des *HMDs* réside dans le fait que le système est peu invasif. Une simple paire de lunettes

autonomes suffit pour profiter d'une visualisation à 360 ° , en relief et en très haute résolution. De leur côté, les *HMDs* sont, pour le moment, forcés d'être reliés par des câbles pour assurer l'alimentation électrique et le transfert des données à afficher. De plus, il y a corrélation entre leur résolution, le poids et l'encombrement de l'appareil. Par contre, les *CAVEs* sont bien plus onéreux que les *HMDs*.

En plus de la vue, les autres sens ont aussi leurs importances dans la sensation d'immersion dans les mondes virtuels, surtout le toucher. De nombreux systèmes ont été développés pour procurer une sensation de toucher :

- soit par retour haptique, avec les systèmes à retour d'effort, souvent couplés aux systèmes d'interactions,
- soit par retour pseudo-haptique, en donnant une impression de contact à travers un autre sens, par exemple, par l'émission d'un son ou la coloration d'un objet lors d'une collision.

2.1.1.3 L'interaction avec les mondes virtuels

L'interaction avec le monde virtuel, ses objets ou les autres avatars qui le peuplent se fait à travers les actions prédéfinies de notre avatar que nous contrôlons à partir d'un périphérique d'entrée. Les périphériques les plus répandus sont le couple clavier-souris ou les différentes manettes de jeux ou joystick disponibles sur le marché. Mais il a été développé une grande variété de périphériques capables de percevoir les actions de l'utilisateur et piloter celles de son avatar. Nous nous intéressons ici plus au moyen technologique de capter de l'information sur les actions effectuées par l'utilisateur qu'aux différents paradigmes d'interaction qui permettent de transformer ces informations en action pour l'avatar.

Une première catégorie de périphériques se base sur le suivi de la position et/ou de l'orientation d'un ou plusieurs points rattachés directement, ou par l'intermédiaire d'un périphérique, à l'utilisateur.

Ces systèmes peuvent s'utiliser sur un contrôleur externe pourvu de capteurs de mouvement comme des gyroscopes, des accéléromètres ou des inclinomètres, comme c'est le cas pour la *Wiimote*® de *Nintendo*® [5] (voir figure 2.4 à gauche). Elle permet d'avoir une information d'orientation absolue et/ou de position relative à une position pré-calibrée.

Ces systèmes peuvent également être mécaniques, s'ils se composent d'un périphérique manipulé par l'utilisateur et relié physiquement à une base fixe. Grâce à des systèmes de codeurs placés aux articulations, il est aisé de retrouver la position et l'orientation du périphérique par rapport à sa base. Le système de ce type le plus connu est le *Phantom*® (voir figure 2.4 au milieu) qui permet de

5. http://fr.wikipedia.org/wiki/Wiimote

manipuler dans l'espace un stylo relié à un bras articulé. Les gants de données, tel le Cyberglove [6] (voir figure 2.4 à droite) se composent, quant à eux, d'un exosquelette relié aux doigts de la main, capable de capter les mouvements de chacune des articulations. Ces systèmes sont également prévus pour appliquer des forces en plus de les capter, ce qui renforce le sentiment d'immersion et de présence grâce au retour haptique.

FIGURE 2.4 – Une *Wiimote*® (gauche), un *Phantom*® (milieu) et un *CyberGlove*® (droite).

D'autres systèmes ponctuels se composent de marqueurs ou cibles placés directement sur l'utilisateur ou sur les accessoires dont on veut assurer le suivi. Ce sont les systèmes de capture du mouvement (*MoCap* pour *motion capture* en anglais). Dans ce cas, différentes technologies peuvent être utilisées. Les systèmes basés sur la force électromagnétique, comme le *Flock of Birds*® [7] ont des marqueurs qui émettent des ondes ou des champs magnétiques qui sont captés par une centrale qui calcule la position de ces marqueurs relativement à sa position. Bien que très précis, ces systèmes ont le désavantage d'être très sensibles aux masses métalliques ou à tout appareil susceptible de générer des champs magnétiques, et d'être eux-mêmes perturbateurs. D'autres systèmes basés sur des marqueurs existent, comme les systèmes comportant des centrales d'inertie tel Intersense [8] ou Xsens [9] (voir figure 2.5 à gauche).

Les systèmes optiques se basent quant à eux sur des systèmes multi-caméra, que nous passerons en revue dans la section §2.1.2. Ils sont surtout utilisés dans l'industrie cinématographique, pour faire piloter des modèles virtuels, créés par des infographistes, par des acteurs réels (voir figure 2.6). Mais ils peuvent être également utilisés pour des applications interactives. Bien souvent, ils sont basés sur un système de caméras pourvues de filtres infrarouges et dotés d'un flash émettant dans l'infrarouge. Les marqueurs sont dans ce cas passifs, des sphères

6. http://www.cyberglovesystems.com/
7. http://www.ascension-tech.com/realtime/RTflockofBIRDS.php
8. http://www.intersense.com/
9. http://www.xsens.com/

FIGURE 2.5 – Un système *Xsens*® basé sur des centrales d'inertie (gauche), et deux systèmes *A.R.T.*® basé sur des marqueurs passifs, l'un pour le suivi du corps entier (milieu) et l'autre pour le suivi de la main (droite).

recouvertes d'une surface réfléchissante, comme dans les systèmes *A.R.T.*® [10] (voir la figure 2.5 au milieu).

D'autres technologies utilisent des marqueurs actifs, des LEDs qui émettent directement à la bonne longueur d'onde (voir la figure 2.5 à droite), comme dans les systèmes *Vicon*® [11]. Ces systèmes sont classiquement utilisés pour le suivi du point de vue de l'utilisateur dans les environnements de visualisation stéréoscopique à la première personne. Il suffit en effet d'équiper la tête de la personne d'une cible comportant plusieurs marqueurs. Nous utiliserons cette technologie dans le chapitre §4. Ce type de système permet d'accéder à une information de position, de vitesse, d'accélération, de suivi et d'identification pour chaque marqueur.

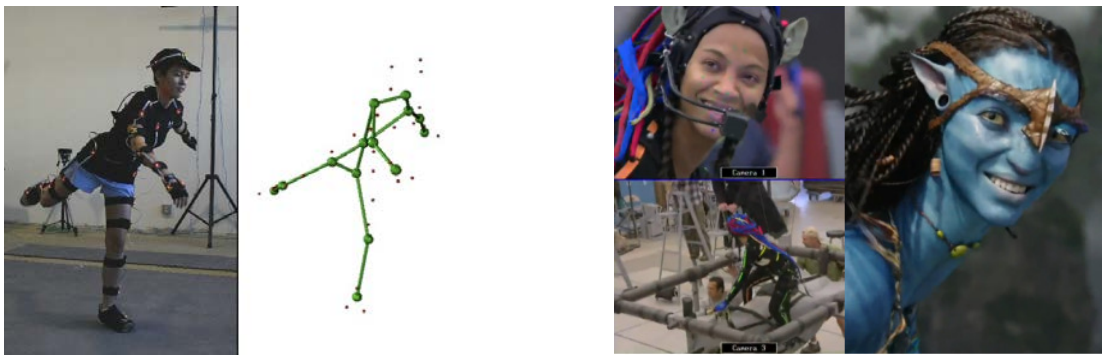


FIGURE 2.6 – Exemples d'utilisation d'un système de capture du mouvement : dans le cadre d'une extraction de squelette en temps-réel (gauche), et dans le cadre du tournage du film *Avatar* (droite).

Tous ces systèmes basés sur le suivi de la position et/ou de l'orientation d'un ou plusieurs points physiques rattachés à l'utilisateur sont suffisamment précis et fiables. ils permettent de piloter des applications ou des avatars en temps-réel. Ils ont, par contre, le désavantage d'être très invasifs, il faut s'équiper au préalable

10. http://www.ar-tracking.de/
11. http://www.vicon.com

et, plus le système comporte de marqueurs, plus la quantité d'information est importante, mais aussi plus la préparation est longue. Il faut également, dans certain cas, calibrer le système en identifiant les marqueurs et en les reliant à un modèle de squelette.

2.1.2 Systèmes de vision

L'utilisation de caméras permet aussi de capter de l'information sur l'utilisateur, plus dense que quelques marqueurs ponctuels, sans pour autant l'obliger à porter des périphériques ou des marqueurs encombrants. Plusieurs travaux se focalisent sur l'obtention de cette information à partir des images, fournies par une ou plusieurs caméras. Ils utilisent pour cela des algorithmes issus du domaine de la vision par ordinateur.

Ces systèmes diffèrent par leur mode de reproduction de l'environnement :

1. monoculaire : reproduction 2D,
2. carte de profondeur par stéréovision, temps-de-vol ou lumière, structurée : reproduction 2.5D ou 2D + profondeur,
3. interpolation multi-vue : reproduction Multi-2D,
4. reconstruction multi-vue : reproduction 3D.

2.1.2.1 Monoculaire

Les premiers systèmes de vision par ordinateur ne comportaient qu'une seule caméra, d'où leur nom de système monoculaire. Ils permettent d'obtenir une image en 2D de la scène observée. De nombreux travaux en Interface Homme-Machine ou en Réalité Augmentée se sont concentrés sur l'utilisation de cette information en 2D afin de déduire plus d'informations sur l'utilisateur, celles-ci peuvent être utilisées à des fins de visualisation ou encore d'interaction.

Le système monoculaire le plus connu est la webcam, présente sur la plupart des ordinateurs portables récents. Celle-ci permet d'avoir une présence visuelle dans les systèmes de vidéo-conférence tels que *Skype*®, qui intègrent déjà des algorithmes de vision par ordinateur comme la détection de fond, afin d'afficher une incrustation de la personne filmée sur un fond de son choix.

L'utilisation à des fins d'interaction des systèmes monoculaire a été initiée par l'équipe de Myron Krueger en 1975 [36] avec *VideoPlace*. L'information de silhouette de l'utilisateur, extraite du flux d'images de la caméra grâce à une installation matérielle, est utilisée pour interagir avec une application en 2D (voir figure 2.7).

FIGURE 2.7 – Example des interactions possibles avec le système *VideoPlace*.

La grande puissance de calcul des processeurs actuels a permis d'adapter d'autres algorithmes de la communauté de la vision par ordinateur et du traitement d'images pour des applications interactives. Le domaine de la réalité augmentée a bien exploré les possibilités de suivi de cibles ou de marqueurs prédéfinis à partir d'une seule caméra (voir ARToolKit [12]) ce qui a permis de créer de nouvelles applications interactives. Mais ces solutions présentent encore le désavantage d'être ponctuelles et basées sur des cibles.

La détection et le suivi d'objets dans les images ont par la suite permis de s'affranchir des marqueurs. La détection, par exemple, des doigts [13, 55], ou encore du visage [15, 84] de l'utilisateur et l'utilisation de leurs positions dans les images rend possible l'interaction avec des objets virtuels.

Les premiers systèmes commerciaux pour les consoles de jeux ont également vu le jour, tel que l'*EyeToy*™ de *Sony Playstation*® [13], qui consiste en une webcam, ou encore la *Wiimote*® de *Nintendo*® [14] (voir figure 2.4 à gauche), manette de jeux qui renferme une caméra infrarouge permettant de détecter des LEDs placées au dessus du téléviseur et donc d'en déduire la position de la manette relativement à l'écran.

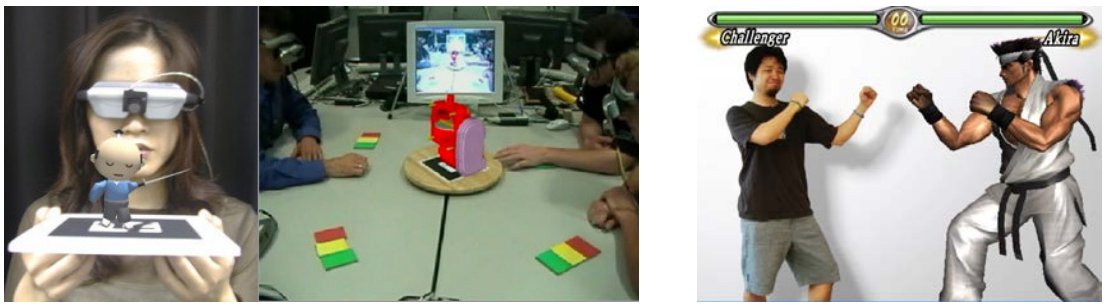


FIGURE 2.8 – Le système *ARToolKit* basé sur la reconnaissance de marqueur et une de ses applications (gauche), et un jeux basé sur la caméra *EyeToy*™ de *Sony Playstation*® (droite).

12. http://www.hitl.washington.edu/artoolkit/
13. http://fr.playstation.com/ps3/peripherals/
14. http://fr.wikipedia.org/wiki/Wiimote

L'utilisation de marqueurs ou de données connues dans la scène permet d'avoir une notion de la profondeur des objets observés dans les images. Il n'en reste pas moins que l'information d'origine étant en 2D, il est difficile d'avoir une connaissance parfaite de l'environnement 3D observé.

2.1.2.2 Stéréovision, caméra temps-de-vol et lumière structurée

D'une seule caméra, les systèmes de vision ont évolués vers deux caméras pour exploiter les possibilités apportées par la stéréovision. L'information recueillie par la paire de caméras est donc composée de deux images 2D qui après étalonnage et traitement donnent une information de profondeur sur la scène observée (carte de profondeur) en plus de l'information de couleur. On appelle cela de la 2,5D. En effet, la connaissance du relief est limitée à ce que voit la paire de caméras, il est donc impossible de connaître l'épaisseur des objets ou de voir les surfaces cachées par un objet au premier plan. Un état de l'art approfondi sur la stéréovision binoculaire a été effectué par Scharstein et Szeliski [63].

Ces systèmes sont aujourd'hui principalement utilisés pour la production cinématographique, surtout avec l'essor de la télévision et du cinéma en relief. Toutefois, les puissances de calcul des nouveaux processeurs et des cartes graphiques récentes rendent possible une extraction de la carte de profondeur en temps-réel. Ceci permet de transmettre sur le réseau une information en relief et de créer des applications de téléprésence où, sous réserve d'être équipé du périphérique d'affichage adapté, on peut apparaître aux autres en relief. Cette information de profondeur permet également d'améliorer les algorithmes de reconnaissance de forme existants pour les systèmes monoculaires [82].

Plus récemment, un autre genre de caméra a fait son apparition sur le marché. Il s'agit des caméra temps-de-vol (*time-of-flight camera* en anglais ou encore *Z-cam*). Ces caméras permettent de récupérer directement une carte de profondeur. Il devient par exemple, aisé de séparer une personne du fond, ce qui n'est pas immédiat avec une seule caméra. Plusieurs travaux utilisent ces informations pour faire de la reconnaissance de visage ou de gestes et du suivi de personne. Ces informations extraites de la carte de profondeur permettent ensuite d'interagir avec une application. Une évaluation de l'utilisation de ces caméras dans le domaine de l'informatique graphique a été effectuée par Kolb et al. [35].

Certains systèmes, dits actifs, projettent des motifs sur l'objet à modéliser. Ils cherchent ensuite à détecter ces motifs dans les images pour reconstruire en 3D la surface sur laquelle les motifs se sont projetés. C'est l'approche dite de la *lumière structurée* (voir figure 2.9). Bien que ces derniers systèmes puissent fonctionner en temps-réel, il n'en reste pas moins qu'ils ne sont pas très agréables

d'utilisation, car la projection tend à éblouir l'utilisateur. Ils sont surtout utilisés dans les scanners 3D pour modéliser des objets fixes.

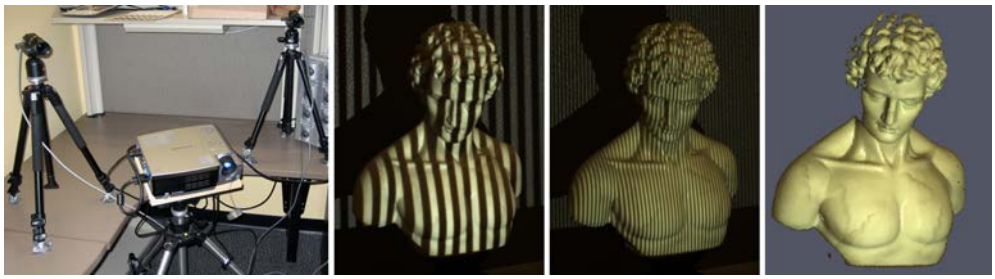


FIGURE 2.9 – Exemple de reconstruction 3D basée sur le concept de la lumière structurée (Université de Brown)

Plus récemment *Microsoft*® a développé un nouveau périphérique, basé sur le principe de la lumière structurée, pour son projet appelé *Kinect*™ [15] (ex *Natal*) qui permet de se passer de manette pour jouer.

FIGURE 2.10 – Une caméra temps-de-vol (gauche), une application de détection de main utilisant cette technologie (milieu), et un jeux basé sur la caméra *Kinect*™ de *Microsoft*® (droite).

2.1.2.3 Multi-vue

La création des premiers systèmes multi-caméra a permis de mettre en avant les problèmes inhérents à l'utilisation de plusieurs sources de données simultanées, comme l'acquisition synchrone des images, le traitement en parallèle des données ou encore l'étalonnage des caméras les unes par rapport aux autres.

Les premiers systèmes multi-caméra créés étaient destinés à l'acquisition *hors-ligne*, c'est à dire à l'enregistrement des flux synchronisés en sortie des caméras pour un traitement ultérieur. Ils ne sont donc pas interactifs et n'ont pas à se soucier des contraintes de performance.

15. www.xbox.com/fr-fr/kinect

Le premier système, *Virtualized Reality*, élaboré par Kanade et al. [33] (voir figure 2.11) se présente sous la forme d'un dôme équipé de 49 caméras, capable d'enregistrer simultanément les images acquises sur des disques et de traiter ces données a posteriori. D'autres systèmes *hors-ligne* ont ensuite vu le jour comme celui de la BBC [25] ou encore celui proposé par Hilton et al. [29].

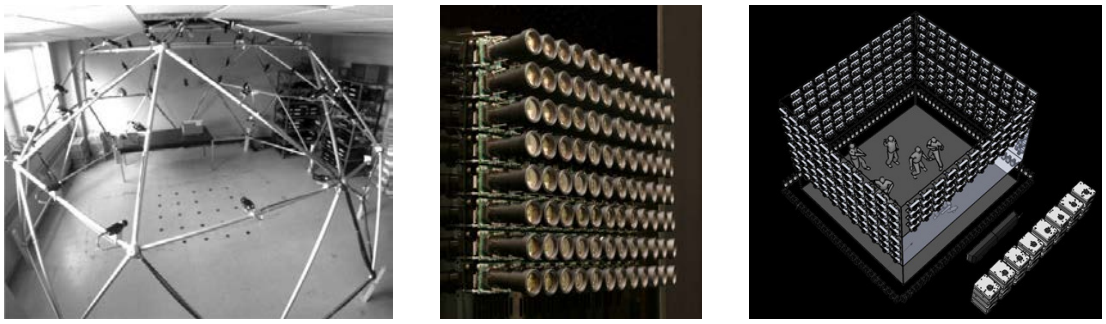


FIGURE 2.11 – Trois systèmes multi-caméra différents : *Virtualized Reality* (gauche), une grille de caméra (milieu), et un concept de super senseur composé de milles caméras (droite).

Ces systèmes vont permettre d'extraire des informations 3D sur l'utilisateur et donc d'avoir des informations qui pourront servir pour de la visualisation à distance ou de l'interaction avec des objets virtuels. Ayant déjà passé en revue les systèmes optiques ponctuels, nous nous intéresserons dans cette partie aux systèmes qui permettent de récupérer une information dense.

Une première catégorie de système a pour but de ne récupérer qu'une information utilisée pour de la visualisation. Il s'agit, à partir du flux d'images prises par les caméras, de pouvoir générer une image de l'objet filmé suivant n'importe quel point de vue. C'est la technique du *point de vue libre* (*free-viewpoint video* en anglais). En utilisant des techniques d'interpolation d'images [10], on peut construire la vue d'une caméra virtuelle à partir des images des caméras ayant un point de vue proche de celui désiré. Zitnick et al. [87] ont développé un système permettant de générer des vues virtuelles en haute définition et cela en temps-réel. Ces techniques sont communément utilisées dans le monde du cinéma car elle permettent de produire le *Bullet Time Effect*, rendu populaire par le film *Matrix* (voir figure 2.12).

Le gros inconvénient de ces systèmes, qui ne génèrent que des points de vue pour de la visualisation, est qu'il n'y a pas de modélisation 3D des personnes présentes dans la scène. Ils sont tout à fait appropriés pour avoir une représentation de soi à 360 ° dans des systèmes de téléprésence mais ils ne sont, d'une part, pas optimaux pour mixer la visualisation de l'utilisateur réel avec des objets virtuels et, d'autre part, ils ne présentent pas d'intérêt pour l'interaction, l'information extraite n'étant que visuelle.

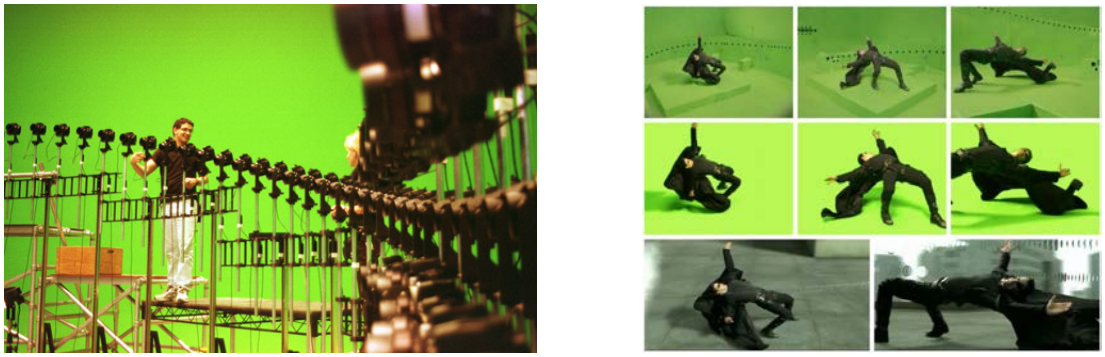


FIGURE 2.12 – L'effet *Bullet Time* dans le filme *Matrix* : à gauche le système multi-caméra, à droite les images acquises et l'incrustation dans le film.

L'autre catégorie de systèmes permet de récupérer une information tri-dimensionnelle sur l'utilisateur présent dans l'espace d'acquisition. Ils vont pour cela utiliser les informations disponibles dans les images prises par les caméras.

Ces systèmes utilisent l'information photométrique disponible dans les images. Ils mettent en correspondance des ensembles de pixels entre les différentes vues et utilisent les mêmes techniques que celles utilisées avec 2 caméras : la reconstruction 3D par stéréovision. Par triangulation, ils récupèrent les coordonnées 3D des zones d'images mises en correspondance, dans le cas de plusieurs caméras, on appelle cela la *stéréovision multi-vue*. Seitz et al. présentent un état de l'art des avancées récentes dans ce domaine [65]. Cette méthode de reconstruction 3D est très coûteuse et difficilement réalisable en temps-réel sans faire de compromis sur la qualité du modèle.

Enfin, le dernier genre de système de reconstruction 3D (que nous étudierons dans le cadre de cet état de l'art), utilise les silhouettes des objets à modéliser dans les différentes images. On peut en effet calculer la géométrie d'un objet à partir de ses silhouettes observées suivant plusieurs points de vue. Laurentini appelle *enveloppe visuelle* [39] l'enveloppe 3D maximale des objets à modéliser dont la projection selon les différents points de vue est à l'intérieur de toutes les silhouettes. L'objet à modéliser est donc forcément inclus dans l'enveloppe visuelle. Le modèle obtenu dépend du nombre de points de vue et n'est qu'une approximation de l'objet réel, il est, par exemple, impossible de modéliser les concavités.

Une première approche, souvent utilisée pour calculer l'enveloppe visuelle, se base sur une modélisation volumique en discrétisant l'espace en *voxels* et en cherchant l'occupation de chacune des zones créées. L'espace peut être partitioné à l'aide d'un *octree* [74, 11]. Une zone fait partie du modèle final si, et seulement si, sa projection dans chaque image fait partie de la silhouette. Un état de l'art des méthodes de reconstruction 3D volumétrique d'enveloppe visuelle a été présenté par Slabaugh [68].

FIGURE 2.13 – Exemple de reconstruction 3D volumique par octree : modèle voxélique à gauche, après une étape de *Marching Cube* au milieu, et après lissage à droite.

Le modèle obtenu est donc un ensemble de *voxels* et sa précision dépend du pas de la grille utilisée, ce qui influence également le temps de calcul. Le rendu visuel est peu satisfaisant à cause de la difficulté à texturer le modèle voxélique à l'aide des données photométriques issues des images d'origine. Une étape supplémentaire de *Marching Cube* peut être appliquée afin de calculer une surface à partir du volume [43] (voir figure 2.13). Cette approche est très utilisée dans un contexte interactif grâce aux implémentations parallèles [69, 66] qui permettent une exécution temps-réel. Elles peuvent donc assurer une présence mécanique à l'utilisateur [28, 48] et lui permettre d'interagir avec des objets virtuels.

Un autre type d'approche de calcul d'enveloppe visuelle se base sur une modélisation surfacique. Elle utilise cette fois-ci non plus l'intérieur des silhouettes mais ses contours afin de déterminer la surface de l'enveloppe visuelle. Cette approche étant celle que nous utilisons avec notre système, je la développerai dans la section 2.2.2.

Notons également que certains systèmes s'intéressent directement à la reconnaissance des mouvements de l'utilisateur [11, 9]. Ils utilisent par exemple un modèle articulé, ou squelette, qu'ils tentent de mettre en correspondance avec les images ou les silhouettes observées (voir figure 2.14). Ensuite, en utilisant éventuellement un modèle 3D de référence, ils essaient de retrouver la forme de l'utilisateur à chaque itération [50]. Comme pour les systèmes ponctuels, cette approche a le mérite de suivre les mouvements de l'utilisateur dans le temps mais les modèles produits sont imprécis car ils se basent sur un modèle initial. De plus, la plupart du temps, il faut également calibrer le système pour adapter le modèle articulé utilisé à l'utilisateur du moment présent.

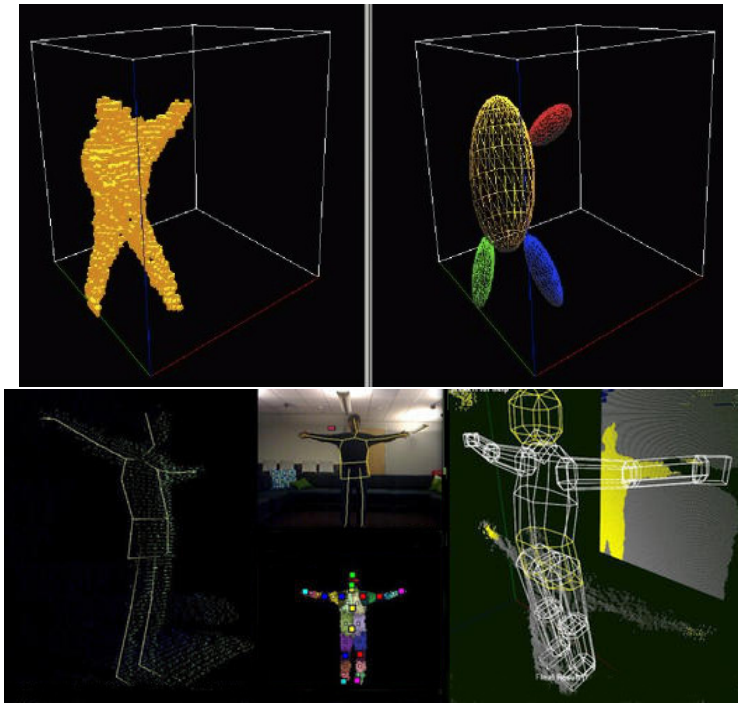


FIGURE 2.14 – Une exemple de mise en correspondance de modèle articulé d'humain sur des données voxéliques (haut), et à partir d'une carte de profondeur et d'une image couleur (bas).

2.2 Contexte : La plateforme *GrImage*

La plateforme *GrImage* est une salle d'expérimentation située à l'*INRIA Grenoble Rhône-Alpes*. Elle comprend un système de vision multi-caméra et un cluster de PC et est utilisée :

- pour l'enregistrement de séquences multi-vue destinées au traitement hors-ligne,
- pour l'acquisition en ligne et la reconstruction 3D temps-réel servant à des applications de réalité virtuelle. C'est cette solution que je développerais.

Dans cette partie, je fais la synthèse de plusieurs publications [1, 3, 59] et énumère le pipeline mis en oeuvre sur la plateforme *GrImage* qui est la base des travaux que je présenterai par la suite.

2.2.1 Un système d'acquisition multi-caméra

Cette partie se concentre sur les caractéristiques techniques nécessaires pour obtenir un flux d'images à partir de plusieurs caméras et pour le transformer en

information utile à l'étape de modélisation 3D, c'est-à-dire sous forme de silhouette calibrée.

2.2.1.1 Acquisition d'images

La méthode de reconstruction 3D que nous utilisons est basée sur des images. Nous devons donc acquérir des flux vidéos à partir de nos caméras digitales. La qualité des images acquises a un impact direct et important sur la qualité du modèle 3D produit. Les webcams disponibles aujourd'hui n'ont pas une qualité suffisante. Ceci est essentiellement dû à leur basse résolution et leur faible taux de rafraîchissement, ou plus important, à leur forte distorsion optique. Ceci les rend inutilisables pour nos applications. Nous utilisons donc des caméras firewire de milieu de gamme pouvant acquérir des images couleurs allant jusqu'à 2 MPixels et ce jusqu'à 60 fois par seconde. Notre plateforme d'acquisition temps-réel est équipée de 8 caméras fixes, chaque caméra est reliée à un noeud de notre cluster dédié au traitement d'images. Nous utilisons une librairie pour contrôler les paramètres des caméras et l'acquisition d'images sous Linux.

Les caméras sont placées de façon à entourer la scène. Le nombre de caméras dépend de la taille de la scène et du niveau de complexité des objets à modéliser ainsi que de la qualité des images requises pour texturer le modèle 3D. L'augmentation du nombre de caméras soulève deux problèmes :

- les imprécisions numériques doivent être gérées avec attention afin de garantir que la qualité du modèle 3D augmente avec le nombre de caméras,
- la complexité du calcul du modèle 3D augmente quadratiquement avec le nombre de caméras.

Au dessus d'un certain nombre de caméras, le modèle obtenu ne s'améliore pas vraiment tandis que la charge réseau augmente ce qui entraîne des latences plus grandes. Nos expériences ont montrées qu'avec 8 caméras nous avions un bon compromis entre la précision, la qualité du modèle et la charge du réseau et des processeurs pour les applications interactives. Pour de l'enregistrement hors-ligne, il est possible d'augmenter leur nombre jusqu'à 32 caméras, la limite est dans ce cas le nombre de cartes d'acquisition et de disques disponibles sur le cluster.

Le positionnement des caméras est motivé par deux choses :

- l'obtention de textures de bonne qualité,
- l'acquisition correcte des contours de la scène.

Par exemple, une caméra positionnée de façon à regarder la scène d'en haut, est habituellement utilisée pour séparer les bras du corps quand l'utilisateur forme un cercle avec ses bras en se tenant les mains loin devant lui. Des caméras placées horizontalement ne seraient pas capable de voir l'espace vide entre ses bras.

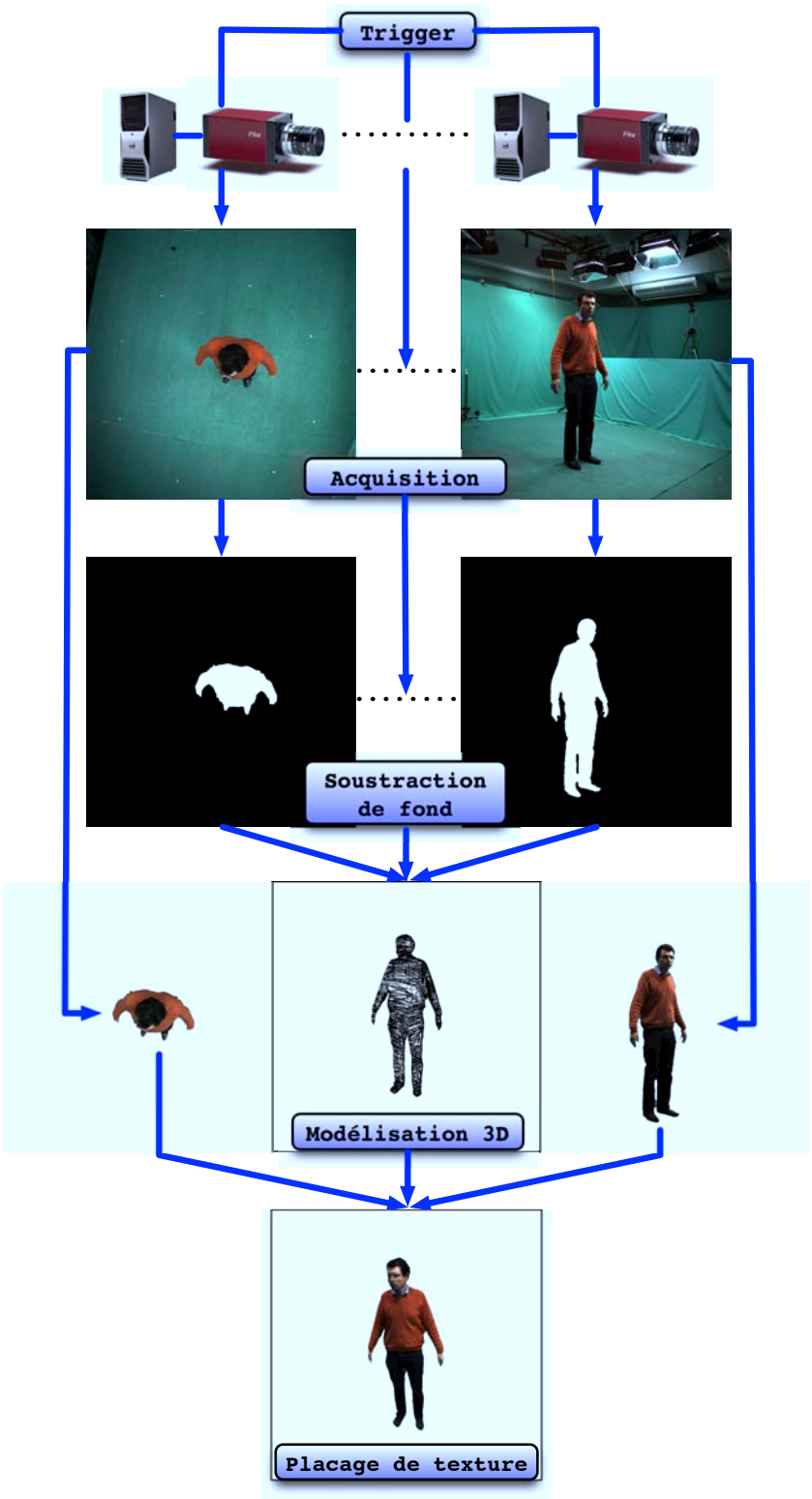


FIGURE 2.15 – Aperçu des étapes menant de l'acquisition d'images à la visualisation d'un modèle 3D reconstruit et des données de textures.

Un bon point de départ est de disposer les caméras uniformément sur une demie sphère délimitant l'espace d'acquisition. La disposition peut être modifiée en fonction de l'application. On peut choisir de placer plus de caméras sur un côté afin d'obtenir une meilleure qualité de texture dans une direction particulière, la face de l'utilisateur, par exemple, ou une meilleure précision du modèle à un endroit, pour les interactions, par exemple.

L'espace d'acquisition, c'est-à-dire l'espace où le modèle 3D peut être calculé, est le volume correspondant à l'intersection des pyramides de vue de chaque caméra. Nous utilisons habituellement deux types d'espace d'acquisition dans nos travaux. Le premier permet à un ou plusieurs utilisateurs d'évoluer dans un espace d'environ trois mètres de diamètre, et l'autre, plus petit, se focalise sur la reconstruction 3D plus précise des mains et forme un cube d'un mètre de côté. La taille de la pièce utilisée pour les expérimentations est, dans notre cas, le facteur qui limite l'expansion de notre espace d'acquisition.

2.2.1.2 Synchronisation

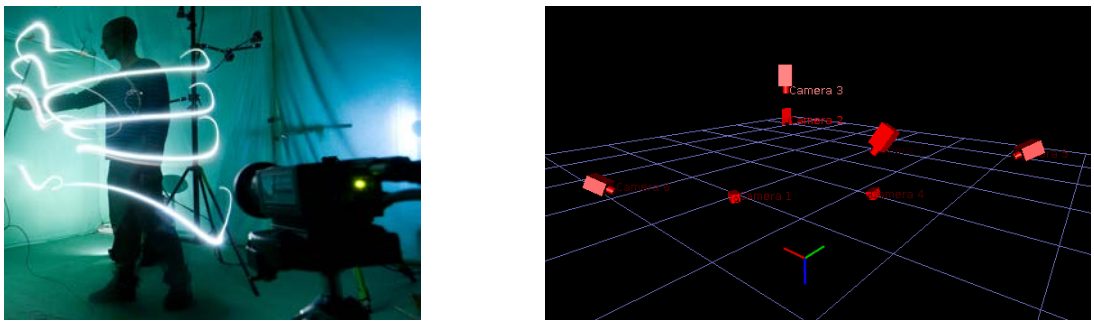
Travailler avec plusieurs sources en entrée pose le problème de la synchronisation des données. En effet, toutes nos applications reposent sur la condition que les images capturées par les différentes caméras soit cohérentes dans le temps, qu'elles reflètent le même événement. Les informations de synchronisation pourraient être récupérées directement depuis les silhouettes en utilisant les inconsistances entre les différents points de vue, comme suggéré dans [67]. Mais une solution matérielle semble plus appropriée et plus efficace dans un environnement dédié comme le nôtre. L'acquisition d'images est donc déclenchée par un signal externe envoyé directement aux caméras via un réseau dédié (*gen-lock* des caméras). Ce mécanisme permet de capturer des vues synchronisées, décalées d'au plus $100\mu s$.

2.2.1.3 Calibrage

Un autre problème posé par l'utilisation de plusieurs caméras est la détermination de leur organisation géométrique dans l'espace. En pratique, nous devons déterminer la position et l'orientation de chacune des caméras dans la scène, ainsi que leurs caractéristiques internes comme la longueur focale. Ceci est réalisé pendant une étape d'étalonnage au cours de laquelle on calcule ces paramètres à partir de la relation entre des points 3D dans la scène et leur projection en 2D dans les images.

Comme pour l'étape de synchronisation, les informations de silhouette pourraient également être utilisées pour récupérer les informations de calibrage en utilisant par exemple [67] ou [7]. Cependant, dans la pratique et pour la précision

des résultats nous préférons utiliser une technique dédiée à notre environnement. Nous réalisons cette étape à l'aide d'un logiciel, que nous avons développé, basé sur des algorithmes d'étalonnage standards comme [51] et [86]. Le processus de calibrage consiste à balayer l'espace 3D avec une baguette sur laquelle quatre lumières, dont la position relative est connue, sont fixées (voir figure 2.16-(a)). Une fois que la position des lumières est détectée et suivie dans le temps et sur toutes les images, une optimisation par *bundle adjustment*, minimise itérativement l'erreur de re-projection dans les images de la position 3D calculée des points lumineux. Ce processus ajuste les paramètres intrinsèques et extrinsèques de chaque caméra à chaque itération de l'optimisation afin de trouver la solution qui produit l'erreur moyenne la plus faible (voir figure 2.16-(b)).



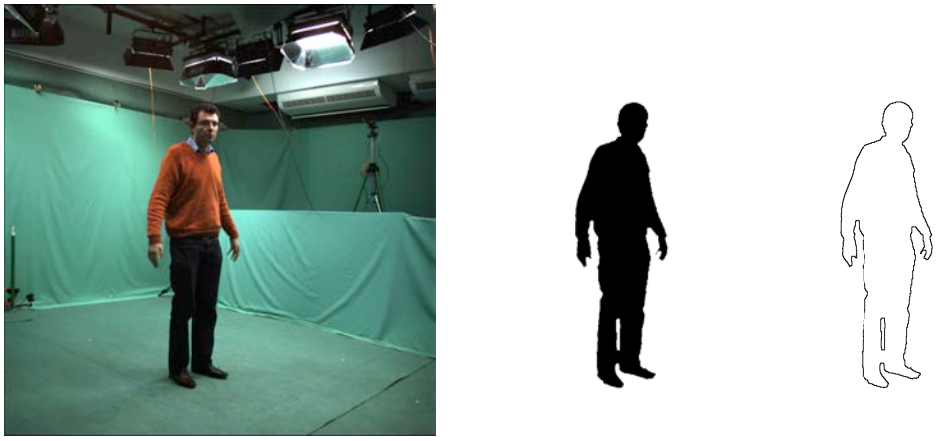
(a) Processus de calibrage (b) Placement des caméras calculé

FIGURE 2.16 – Le calibrage.

2.2.1.4 Soustraction de fond

Les régions d'intérêts dans les images, c'est-à-dire le premier plan ou encore la silhouette, sont extraites lors de l'étape de soustraction de fond. Nous considérons que la scène est composée d'un fond statique que nous avons préalablement appris. Comme la plupart des techniques existantes [31, 11], nous nous basons sur un modèle de fond par pixel. Pour ce modèle, nous utilisons une combinaison de Gaussienne pour les informations chromatiques (UV) et de modèle d'intervalle pour l'information d'intensité (Y) avec une variante de la méthode d'Horprasert et al. [31] pour la détection des ombres. Il est important de noter que la précision du modèle 3D produit dépend beaucoup de cette étape.

Une soustraction de fond de qualité peut être facilement réalisée avec un environnement dédié comme les salles vertes/bleues dites *chromakey*. Bien que nous utilisons également majoritairement ces couleurs, car elles se différencient bien de la couleur de la peau et des vêtements usuels, notre méthode ne se limite pas à des environnements aussi spécifiques.



(a) (b) (c)

FIGURE 2.17 – Les différentes étapes du traitement d'images : l'acquisition d'images (a), la soustraction de fond (masque binaire) (b), la polygonalisation de la silhouette (250 sommets).

2.2.1.5 Polygonalisation des silhouettes

Comme notre algorithme de reconstruction 3D produit une surface et non un volume, il n'utilise pas les régions d'images définies par les silhouettes mais leurs contours extérieurs. Nous extrayons ces contours de silhouettes en les vectorisant avec la méthode de Debled-Rennesson et al. [16]. Chaque contour est décomposé en un polygone orienté, qui estime le contour jusqu'à une limite donnée. Avec une limite de un pixel, les polygones obtenus sont équivalents strictement aux contours des silhouettes. Mais, dans le cas de silhouettes bruitées, cela crée un nombre important de petits segments qu'une limite d'approximation plus grande permet de supprimer. Cette limite offre la possibilité de contrôler la complexité du modèle 3D produit, et donc du temps de calcul nécessaire à sa production de façon efficace.

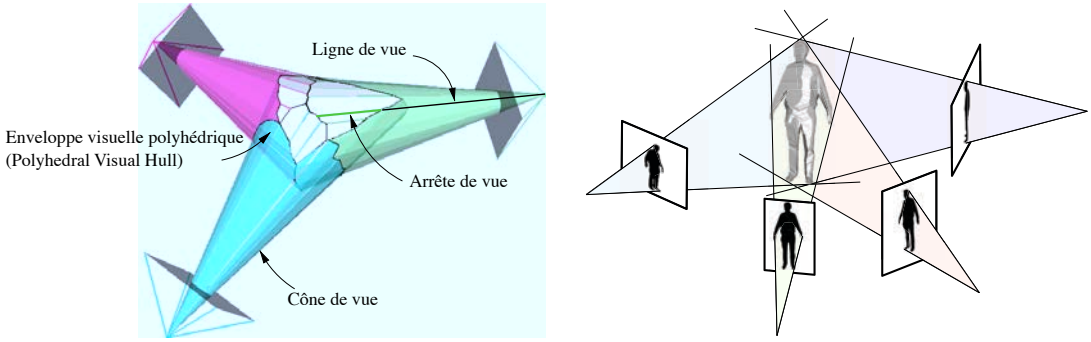
2.2.2 Reconstruction 3D temps-réel

Pour obtenir un modèle 3D géométrique des objets ou personnes présents dans l'espace d'acquisition, nous utilisons une méthode basée sur les silhouettes extraites des images, qui construit un modèle de forme appelé l'*enveloppe visuelle*. Les méthodes de modélisation des formes à partir des silhouettes, présentées dans la section §2.1.2.3, sont bien adaptées à notre contexte car elles présentent l'avantage d'être exécutables en temps-réel. D'autres approches existent, comme [65, 78, 22], mais la plupart n'arrivent pas à produire un modèle 3D en temps-réel pendant

une longue période et avec une robustesse et une efficacité équivalente à celle des approches basées sur les enveloppes visuelles.

2.2.2.1 L'enveloppe visuelle

L'enveloppe visuelle est une forme géométrique très étudiée [39, 40] qui est obtenue à partir des n silhouettes générées par les objets présents dans la scène. Géométriquement, l'enveloppe visuelle est l'intersection des *cônes de vue*, le sommet de ces cônes est le centre optique de la caméra et leurs supports correspondent aux contours des silhouettes. Quand on considère des contours de silhouettes polygonalisés, l'enveloppe visuelle devient un polyèdre régulier.



(a) (b)

FIGURE 2.18 – Principe de l'enveloppe visuelle.

Notre travail est basé sur l'algorithme *EPVH* [19, 20] (pour *Exact Polyhedron Visual Hull*). Cet algorithme a comme particularité de construire un modèle 3D exact, c'est-à-dire dont la re-projection dans les images coïncide exactement avec les silhouettes observées. Ceci est intéressant quand le modèle nécessite d'être texturé, car cela permet aux textures extraites depuis les silhouettes de pouvoir être appliquées directement sur le modèle.

La méthode que nous utilisons ici récupère l'enveloppe visuelle de l'objet de la scène sous la forme d'un polyèdre. Comme expliqué précédemment, nous récupérons les contours de la silhouette de chaque image comme des polygones 2D. Une description polygonale discrète des silhouettes implique qu'il n'y ait qu'une unique représentation de l'enveloppe visuelle. Pour obtenir cette représentation trois étapes sont nécessaires :

- Un sous-ensemble des arêtes du polyèdre est calculé : les arêtes de vue.
- Le reste des arêtes du polyèdre est récupéré par une série de déduction géométrique récursive. La position des sommets n'ayant pas encore été calculée est graduellement déduite de ceux déjà obtenus, en utilisant les arêtes de vue comme condition initiale.

- Finalement, le maillage est parcouru pour identifier les facettes du polyèdre.

Voici un rapide aperçu de l'algorithme séquentiel. Pour plus de détails, se reporter à [20]. Ensuite je présenterai la version distribuée, créée pour atteindre des performances temps-réel.

2.2.2.2 Calculer les arêtes de vue

Les arêtes de vue sont les arêtes de l'enveloppe visuelle induites par les lignes de vue qui partent des sommets des contours des silhouettes (voir figure 2.19). Il y a une ligne de vue par sommet 2D. Sur chaque ligne de vue, EPVH identifie les segments qui se projettent à l'intérieur des silhouettes de toutes les autres images. Chaque segment, appelé arête de vue, est une arête de l'enveloppe visuelle et chaque extrémité des segments en est un sommet 3D. Chaque sommet 3D est trivalent, c'est à dire qu'il est l'intersection de 3 arêtes.

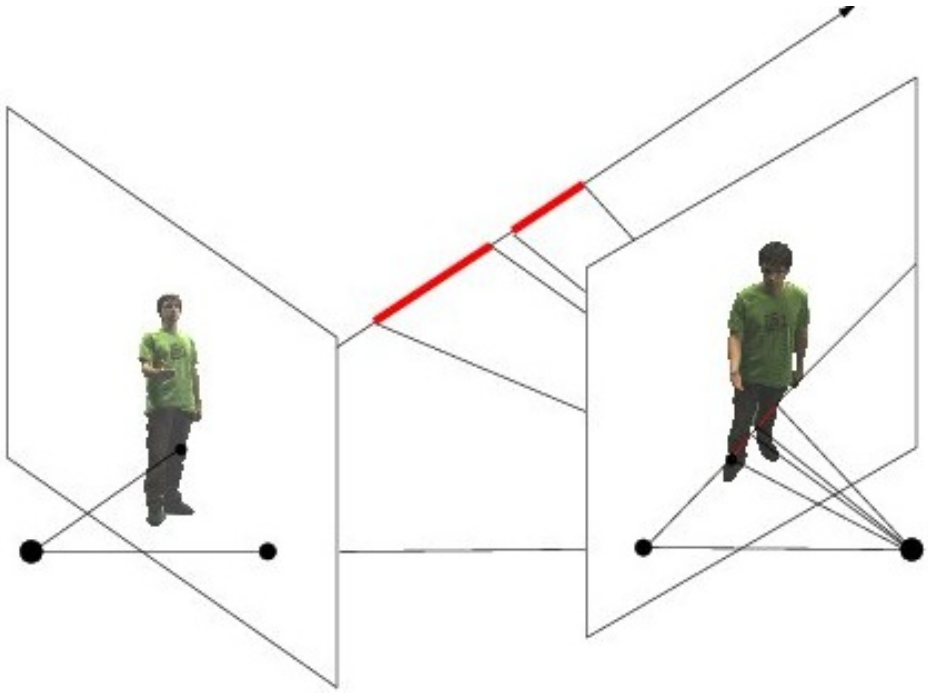


FIGURE 2.19 – Arêtes de vue (en gras) le long de la ligne de vue.

Pour chaque arête, EPVH garde les deux plans générateurs, c'est à dire les deux plans définis par la projection des deux arêtes de la silhouette incidentes au sommet 2D utilisé pour créer la ligne de vue. EPVH garde également pour chaque *sommet de vue*, le troisième plan générateur, défini par la projection de l'arête de la silhouette qui intersecte la ligne de vue.

2.2.2.3 Calculer le maillage de l'enveloppe visuelle

Après cette première étape, l'enveloppe visuelle n'est pas encore complète. Certaines arêtes manquent pour remplir les conditions de connectivité du maillage (il doit être manifold). Certains sommets, appelés *triple points*, manquent également. Un *triple point* est un sommet de l'enveloppe visuelle généré par l'intersection des trois plans définis par les segments de silhouette de trois différentes images.

EPVH complète le maillage en évoluant le long des arêtes 3D définies par deux arêtes de silhouette, aussi longtemps que ces arêtes 3D se projettent à l'intérieur de toutes les silhouettes. Aux limites, soit quand une arête se projette sur un contour de silhouette, EPVH identifie de nouveaux *triple points*, ou reconnait un sommet déjà calculé.

Les sommets existants déjà sont seulement connectés par une arête, une arête de vue. Cette arête est délimitée par l'intersection de deux plans générateurs qui appartiennent à la même silhouette. EPVH intersecte chacun de ces plans générateurs avec le troisième, pour créer l'arête manquante. Considérons P comme le troisième plan générateur. L'extrémité manquante de chacune de ses arêtes est tout d'abord délimitée par un sommet existant, celui défini par l'intersection avec un des plans générateurs incidents à P dans la même silhouette. Puis, de la même façon que pour les arêtes de vue, EPVH vérifie que ce segment croise bien les autres silhouettes. Dans ce cas, il crée un nouveau *triple point*. Ce processus nous donne un maillage complètement connecté.

2.2.2.4 Calculer les facettes

Une dernière étape consiste à traverser le maillage pour identifier les facettes du polyèdre. Les contours des facettes 3D sont extraits en parcourant le maillage orienté et en prenant à gauche à chaque sommet. L'information d'orientation est déduite de l'orientation des silhouettes (sens inverse des aiguilles d'une montre pour les contours externes et sens horaire pour les contours internes).

2.2.3 Algorithme distribué

2.2.3.1 Algorithme

Afin d'atteindre des performances temps-réel, une version parallèle de l'algorithme EPVH a été développée. Elle se décompose en un pipeline à 3 étages :

- **Étage 1 : Arêtes de vue**. Soit V le nombre de *thread* (fil d'exécution ou processus léger) – chaque *thread* est distribué sur un CPU différent du cluster – en charge de calculer les arêtes de vue. Les silhouettes extraites par tous les hôtes en charge du traitement d'images sont diffusées aux V *threads*. Chaque *thread* calcule localement les arêtes de vue correspondantes aux n/V lignes de vue, où n correspond au nombre total de lignes de vue (voir figure 2.20-(a)).
- **Étage 2 : Connectivité du maillage**. Soit M le nombre de *threads* en charge de calculer le maillage. Les V *threads* de l'étage précédent diffusent les arêtes de vue aux M *threads* de cet étage. Chaque *thread* se charge d'une *tranche* de l'espace (découpée selon l'axe vertical comme nous avons principalement à faire à des humains debouts) où il calcule le maillage. Les *tranches* sont définies de façon à contenir le même nombre de sommets. Chaque *thread* complète la connectivité de son *sous-maillage*, en créant des *points triples* quand c'est nécessaire. Les *sous-maillages* sont ensuite rassemblés sur un des hôtes qui fusionne les résultats en prêtant attention à la connectivité sur les frontières des *tranches*, et rajoute des arêtes ou des sommets manquants (voir figure 2.20-(b)).
- **Étage 3 : Identification des facettes**. Le maillage est ensuite diffusé aux K *threads* en charge de l'identification des facettes. La charge de travail est partagée par distribution des plans générateurs sur les différents processeurs (voir figure 2.20-(c)).

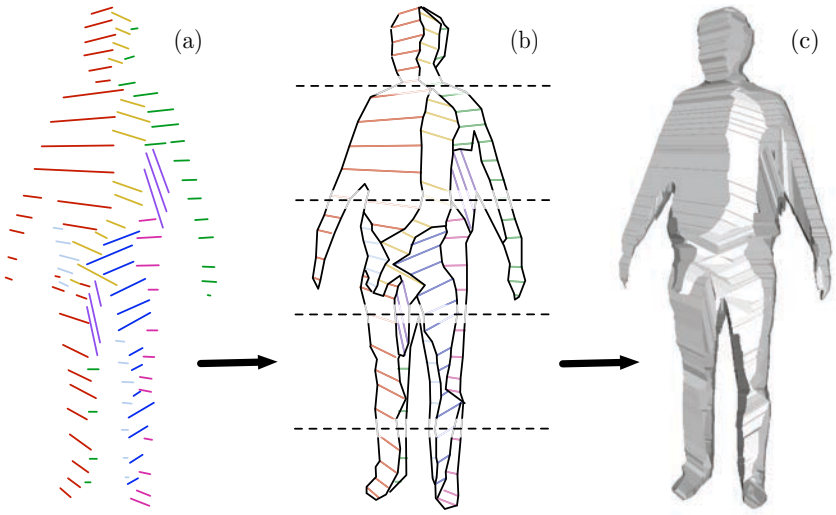


FIGURE 2.20 – Les trois étapes principales de l'algorithme parallèle EPVH. (a) Calcul des arêtes de vue. (b) Calcul de la connectivité du maillage (les coupes horizontales représentent le partitionnement utilisé pour la parallélisation. (c) Génération des facettes.

2.2.3.2 Évaluation

On considère un espace d'acquisition couvert par au maximum 8 caméras et avec une ou deux personnes à modéliser. Dans ce cas, l'algorithme tourne à la même fréquence que le taux de rafraîchissement des caméras (30 images/s) et assure une latence inférieure à 100 ms (acquisition et traitement d'images inclus) en utilisant 4 à 8 processeurs. Bien que l'algorithme soit flexible et permette l'utilisation de plus de processeurs, cela n'entraîne pas de nette amélioration des performances. Plus d'informations sur la parallèlisation de cet algorithme et ses performances peuvent être trouvées dans [21] et [49].

L'utilisation d'un plus grand nombre de caméras implique plusieurs problèmes. La complexité de l'algorithme est quadratique selon le nombre de caméras, ce qui provoque des latences qui ne sont pas acceptables pour des applications interactives. Nous cherchons donc davantage à utiliser des caméras avec une meilleure résolution plutôt qu'à augmenter leur nombre. Utiliser plus de caméras serait utile pour de grands espaces d'acquisition où l'on peut produire des modèles 3D pour chaque sous-ensemble de l'espace. La complexité algorithmique est en $n \log(n)$, où n est le nombre maximum de segments par silhouette, il est donc plus intéressant de réduire la complexité en jouant sur ce paramètre.

2.2.4 Visualisation et interactions

Le maillage et ses textures associées, extraites des images d'origine, sont envoyés sur le réseau aux noeuds de visualisation et de simulation physique qui s'occupent des interactions (voir figure 2.21). Ces étapes sont décrites dans les paragraphes suivants.

2.2.4.1 Simulation

Le couplage de la modélisation 3D temps-réel avec un moteur de simulation physique offre de nouvelles possibilités en termes d'interaction : des interactions non symboliques qui semblent plus naturelles à l'utilisateur. Grâce au logiciel *SOFA* [16], nous avons développé un simulateur distribué qui gère les collisions entre des objets déformables ou non et le corps de l'utilisateur modélisé. On peut de se servir de toutes les parties de notre corps ou les accessoires présents dans l'espace d'acquisition, ce qui n'est pas faisable avec les systèmes usuels d'interaction. De plus, cette méthode a l'avantage d'être non invasive. Mais certaines interactions

16. www.sofa-framework.org

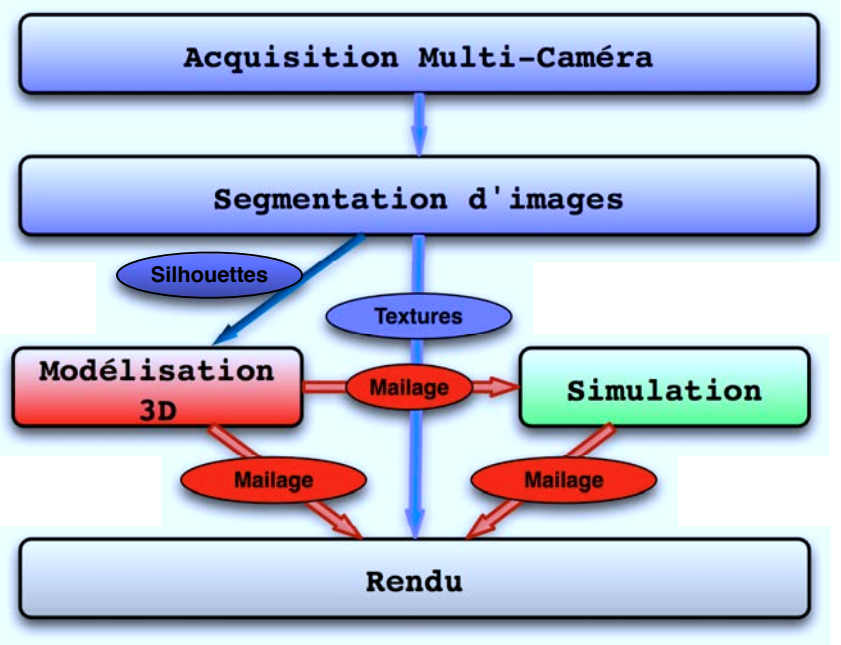


FIGURE 2.21 – Architecture de l'application de couplage d'un système multi-caméra utilisée pour de la reconstruction 3D temps-réel et d'un environnement physique virtuel.

sont compliquées, la préhension d'objets par exemple est très difficile, dans la mesure où il n'y a pas d'information de force liée au modèle 3D.

Le logiciel *SOFA* (pour *Simulation Open Framework Application*) est un logiciel libre qui cible initialement le milieu de la recherche en simulation médicale [3]. L'intégrer dans notre application a juste requis l'ajout d'un composant qui reçoit le flux de maillages 3D issus de la reconstruction 3D et l'associe à un modèle mécanique utilisé pour la détection de collision. Du point de vue de la simulation physique, le modèle 3D est vu comme un corps rigide, insensible aux forces externes (avec une masse infinie) et dont la structure est mise à jour à chaque itération.

Afin d'obtenir des interactions plus précises, la réaction suite à la collision est fonction de la vitesse et de la direction du mouvement à l'endroit de l'impact. De cette façon, l'utilisateur peut pousser des objets virtuels, par exemple, tirer dans un ballon, au lieu de juste les bloquer. Actuellement, nous calculons cette vitesse en cherchant la distance minimum entre le point de la surface à l'endroit de la collision et le maillage à l'itération précédente. Cette information de vitesse est donc une approximation du mouvement de l'utilisateur perpendiculaire à la surface, ce qui est suffisant pour produire une force de réaction. Le mouvement tangentiel ne peut pas être calculé de cette façon. On cherchera à trouver une solution à cette limitation dans le chapitre §6.

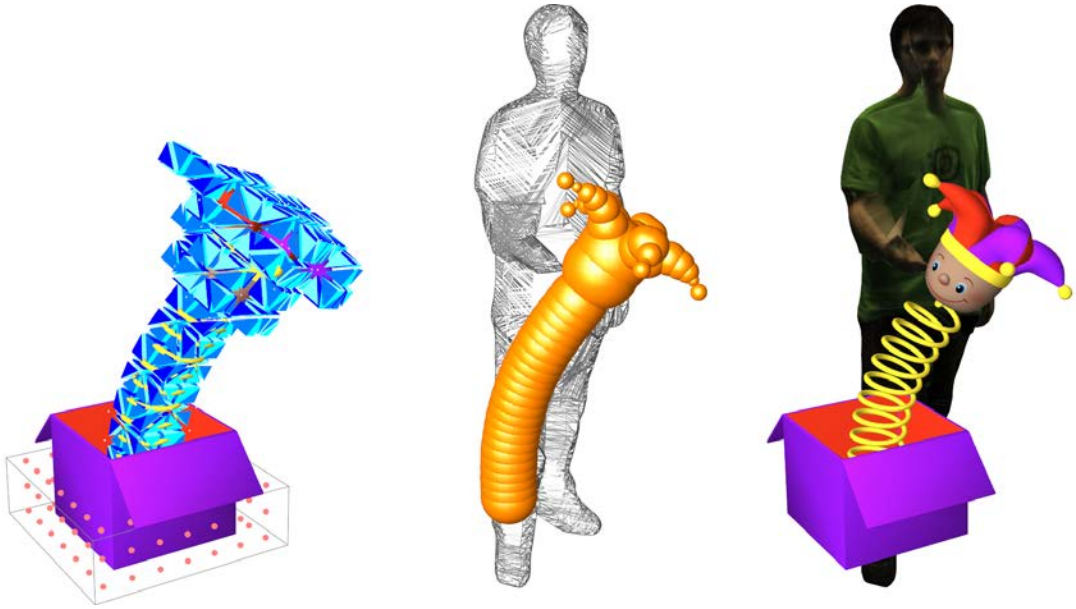


FIGURE 2.22 – Un objet déformable (gauche) entre en collision avec un maillage reconstruit en 3D (milieu) permettant des interactions avec l'utilisateur (droite).

2.2.4.2 Rendu

Les données à rendre, fournies soit par la simulation physique, soit par un chargeur de scène statique, soit par l'algorithme de modélisation 3D, sont distribuées aux noeuds de rendu. Le rendu peut être effectué sur une grande variété de périphériques : du moniteurs standard, aux vidéo-projecteurs stéréoscopiques.

Pour distribuer efficacement la partie rendue, nous utilisons *FlowVR-Render* [4]. Les solutions existantes pour paralléliser le rendu ou effectuer du rendu à distance se basent sur la communication de pixels, de commandes OpenGL ou de changements dans un graphe de scène. Nous comptons sur une solution intermédiaire basée sur des primitives graphiques indépendantes qui utilisent des *shaders* matériels pour spécifier leur apparence visuelle. Comparée à une approche basée sur OpenGL, cette méthode élimine la machine à état qui crée des dépendances entre les données et rend l'ordonnancement des primitives et la fusion des différents flux compliqués.

Le rendu du modèle 3D est assez simple dans la mesure où il s'agit d'une surface polygonale. Pour appliquer les textures extraites des silhouettes, nous utilisons un *shader* qui projette les sommets du maillage dans les images d'origine à l'aide des paramètres d'étalonnage des caméras. L'algorithme EPVH garantit que le modèle 3D peut être re-projeté dans les silhouettes d'origine avec une erreur minimale, une propriété qui permet de réaliser une texture de très bonne qualité. En prenant en compte la normale à la surface, la direction de la caméra OpenGL et les occultations propres, le *pixel shader* combine les contributions des différentes

caméras. L'accès à la totalité du modèle 3D permet un rendu interactif et sans contraintes depuis n'importe quel point de vue.

2.2.5 Implémentation

Le couplage des différents composants logiciels impliqués dans ce projet et la distribution sur les noeuds du cluster de PC, pour atteindre des performances temps-réel à l'exécution, est accompli grâce à l'intergiciel *FlowVR* [2, 42]. Un intergiciel développé à l'*INRIA* conjointement au projet *GrImage*.

FlowVR renforce une programmation modulaire qui permet une exécution performante sur des architectures parallèles ou distribuées. *FlowVR* s'appuie sur un modèle de *flot de données* et sur une approche orientée composant qui a été utilisée avec succès pour d'autres applications de visualisation scientifique. Le développement d'une application *FlowVR* est un processus en deux étapes. Premièrement, on développe des modules. Un module est une boucle sans fin qui consomme des messages sur ses ports d'entrée à chaque itération et produit de nouveaux messages sur ses ports de sortie. Ils encapsulent un bout de code, importé d'une autre application ou développé de zéro. Ce code peut-être parallèle ou *multi-thread*. Dans une seconde étape, les modules sont répartis sur l'architecture cible et assemblés en un réseau qui définit comment les données vont être échangées. Ce réseau peut utiliser des mécanismes avancés comme des filtres de routage intelligent des messages ou des processus de synchronisation entre les flux.

À l'exécution *FlowVR* lance un *démon* sur chacun des noeuds du cluster. Ce *démon* est en charge de la synchronisation et de l'échange des données entre les modules. Ceci cache tous les aspects relatifs au réseau et rend le développement des modules plus simple. Chaque *démon* alloue un segment de mémoire partagé. Les messages utilisés par les modules sont lus et écrits sur ce segment. En cas d'échange de données entre deux modules situés sur le même noeud, un simple passage d'adresse mémoire est effectué. Pour les communications entre deux modules situés sur des noeuds différents, le *démon* se charge des communications distantes sur le réseau.

Les applications *FlowVR* les plus importantes sont composées de centaines de modules et de milliers de connections. Afin d'être capable de gérer la création du réseau de telles applications, *FlowVR* est basé sur un modèle de composants hiérarchiques [41]. Ce modèle introduit un nouveau type de composant appelé *composite*. Un *composite* est créé en assemblant d'autres composants *FlowVR*. Cette hiérarchie permet de créer des batteries de patrons ou squelettes efficaces et réutilisables grâce à des jeux de paramètres.

Le réseau *FlowVR* qui correspond à l'application *GrImage* contient des milliers de modules et connections développés en assemblant des composants ensemble (voir figure 2.23). Le réseau utilisent plusieurs patrons pour gérer les communications. Par exemple, dans le composant en charge de l'acquisition et du traitement d'images, un pipeline est associé à chaque caméra. La reconstruction 3D nécessite une forte cohérence entre ces pipelines, pour cette raison une barrière de synchronisation s'assure que les données en entrée de la modélisation 3D sont bien synchrones. Afin d'atteindre des performances temps-réel, il est parfois nécessaire de supprimer certaines *méta-trames* car le système ne sera pas en mesure de les traiter en temps-réel sans accumuler de la latence. Le patron de synchronisation est donc en charge d'effectuer le ré-échantillonnage des données si nécessaire.

La compilation du réseau *FlowVR* renforce la modularité de l'application. Le description hiérarchique de l'application de *GrImage* est complètement indépendante du type d'architecture ou de cluster utilisé. Un fichier décrit l'architecture et les contraintes matérielles comme les associations caméra-machine et sert à la compilation du réseau. Le bon nombre de pipeline d'acquisition et de traitement d'images sera décidé en fonction du nombre de caméras. Le placement de ces pipelines sera effectué en adéquation avec les branchements physiques des caméras.

De cette façon tout changement matériel effectué, comme l'intégration d'un écran stéréoscopique ou la présence d'un noeud avec plus de coeurs, nécessitera seulement un changement du fichier décrivant l'architecture. Cette modularité est critique surtout dans le cas d'une application comme celle de *GrImage* qui a été développée durant plusieurs années par différentes personnes. *FlowVR* est un composant clef pour rendre possible une telle association de codes informatiques sans compromis sur l'efficacité à l'exécution.

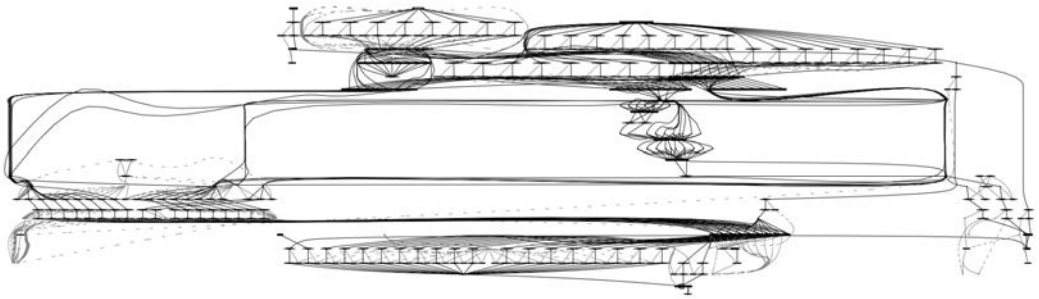


FIGURE 2.23 – Le réseau *FlowVR* de l'application *GrImage*. Les noeuds représentent les modules et les arêtes les communications. Ce réseau est compilé pour 8 caméras et une version d'EPVH parallélisée sur 6 CPUs.

Chapitre 3

Téléprésence

3.1 Motivations et problématiques

Les environnements virtuels ont été initialement conçus pour les jeux multi-joueurs en ligne, mais des applications comme *Second Life* ont montré qu'ils pouvaient très bien être utilisés pour supporter des réseaux sociaux ou des environnements de travail collaboratif distribués (voir figure 3.2 et 3.2). Ces environnements sont les fondations de ce qui pourrait devenir le mode de communication et de collaboration à distance du futur. Mais il manque encore des fonctionnalités pour permettre des interactions riches entre utilisateurs, parmi elles, l'habilité d'assurer un fort sentiment de présence aux utilisateurs et le support des environnements qui nécessitent de grandes masses de données.

La présence de l'utilisateur est aujourd'hui souvent limitée à son avatar contrôlé par un périphérique simple comme un clavier et une souris. Bien que cet avatar puisse être ressemblant, sa capacité à véhiculer des expressions corporelles est

FIGURE 3.1 – Une application commerciale de télé-conférence.

pratiquement inexistante, ce qui dégrade le sentiment de présence de l'utilisateur et de ceux qui interagissent avec lui.

FIGURE 3.2 – Réunion dans l'environnement virtuel *Second Life*.

Les travaux présentés dans la partie précédente permettent d'immerger une personne dans un environnement virtuel local en créant un clone virtuel qui véhicule son image et ses possibilités d'expression et d'interaction. Nous avons décidé d'explorer les possibilités d'utilisation des systèmes de reconstruction 3D multi-caméra comme périphérique d'entrée pour des applications de téléprésence.

Les difficultés induites par l'utilisation de cette technologie dans un contexte de téléprésence et d'interaction à distance sont liées à l'habilité à calculer un modèle de bonne qualité, et à le transmettre sur le réseau avec un taux de rafraîchissement et une latence qui permettent un bon niveau de téléprésence et d'interactivité.

Outre le problème de transfert des données, le développement d'un monde virtuel partagé, où plusieurs utilisateurs peuvent se connecter simultanément, entraîne des problèmes relatifs à l'accès aux données partagées dans un contexte collaboratif, comme l'accès concurrent aux données. Il faut également déterminer un mode de communication entre les utilisateurs, ce mode peut être de type client-serveur, client-à-client ou un mixe des deux selon les données à communiquer.

Le matériel utile à la discussion, ou à la collaboration à distance, peut également nécessiter un jeu de données complexe. Par exemple, une application de coordination de secours, à la suite d'une catastrophe naturelle, nécessite d'avoir accès à plusieurs sources de données, des données pré-existantes sur la topologie du terrain, complétées par des informations recueillies par des capteurs ou des observations de terrain en temps-réel, et éventuellement améliorées par des données issues d'une simulation physique. Ces grands jeux de données doivent donc

également être transférés sur le réseau efficacement et de manière cohérente avec le reste des données.

3.2 Concept

Nous avons donc cherché à développer une application qui permet à plusieurs personnes distantes d'être modélisées en 3D afin d'être présentes dans un monde virtuel commun aussi bien visuellement que mécaniquement. Notre contribution dans ce domaine est démontrer expérimentalement l'intérêt de la reconstruction 3D pour de telles applications de téléprésence. Nous avons créer pour cela une plateforme expérimentale mettant en oeuvre plusieurs systèmes de reconstruction 3D multi-caméra distants (voir figure 3.3).

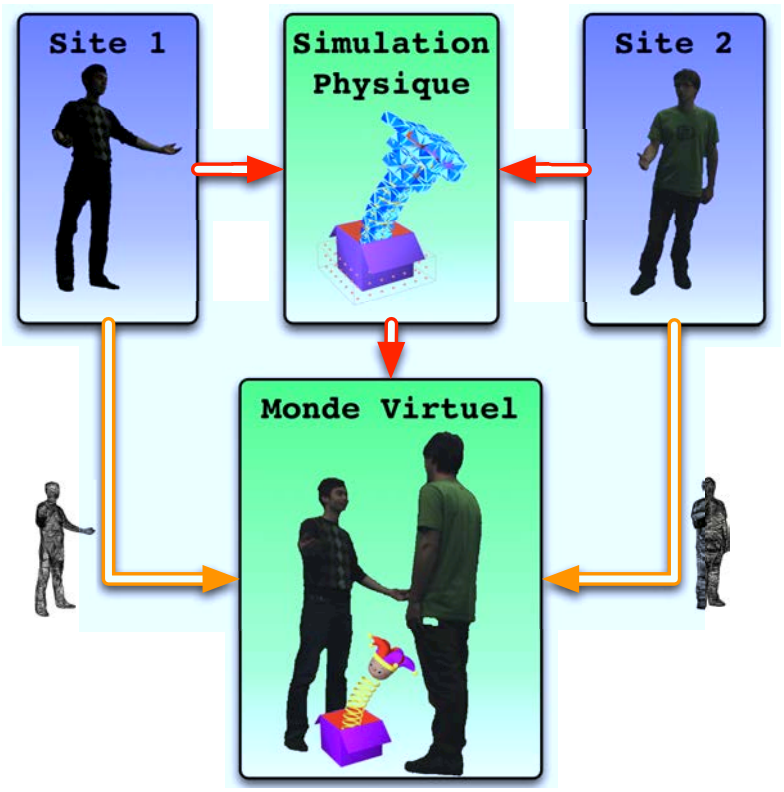


FIGURE 3.3 – Concept d'application de téléprésence et collaboration à distance basée sur des plateformes de reconstruction 3D et une simulation physique.

3.2.1 Présence visuelle

Du point de vue de l'utilisateur, le sentiment de présence est grandement amélioré par l'utilisation d'un avatar qui possède sa forme et son apparence plutôt

que par celle d'un avatar synthétique sélectionné dans une banque de modèle 3D. De plus, cet avatar se déplace suivant ses propres mouvements et non en fonction d'actions pré-programmées et déclenchées au moyen d'un périphérique. Dans un contexte de téléprésence, cela permet à plusieurs utilisateurs de se reconnaître et d'avoir des conversations similaires à celles qu'ils pourraient avoir dans la réalité. Les émotions sont véhiculées au travers des expressions faciales, ou de la gestuelle, ce qui est très difficile à mettre en oeuvre, et forcément limité, avec un avatar synthétique.

3.2.2 Présence mécanique

Partager notre apparence n'est pas le seul avantage des environnements basés sur la reconstruction d'un modèle 3D de l'utilisateur. Les maillages 3D produits peuvent également servir pour interagir avec des objets virtuels dans un contexte partagé tout comme ils le permettent dans un contexte local. Un serveur gérant l'environnement virtuel reçoit les informations mécaniques des utilisateurs (maillage 3D et autres actions sémantiques), les injecte dans le moteur de simulation physique et renvoie les transformations de la scène aux différents utilisateurs. Les objets déformables dynamiquement sont gérés par ce serveur alors que le reste de la scène qui est statique est chargé à l'initialisation par chaque noeud de rendu.

Ce genre d'environnement permet donc à plusieurs utilisateurs, distants physiquement, d'interagir ensemble, avec les mêmes objets virtuels. Il n'y a pas de problèmes d'accès concurrent aux données. À chaque pas d'itération, le serveur de simulation calcule les collisions entre les objets virtuels et les maillages des différents utilisateurs. De ces collisions, il déduit le changement d'état à appliquer aux objets. Comme dans la réalité, si deux personnes poussent un objet dans le sens opposé, l'objet aura tendance à ne pas bouger. La simulation physique n'ayant pas d'information de force, elle ne pourra pas favoriser une collision plutôt qu'une autre.

Il est bien sûr impossible de modifier l'état des utilisateurs vu qu'il ne sont pas équipés de périphérique à retour d'effort. De même qu'il est impossible à deux participants d'interagir mécaniquement l'un avec l'autre. Ces systèmes pourraient par contre profiter d'un périphérique à retour d'effort pour améliorer la sensation d'interaction.

3.3 Travaux connexes

Les systèmes de téléprésence basés sur des réseaux de caméras garantissent un niveau de présence différent, qui dépend essentiellement du nombre de caméras et de l'algorithme de reconstruction 3D utilisé. Il existe en effet plusieurs techniques pour calculer le modèle 3D d'une personne à partir d'un flux d'images (voir section §2.1).

Certains systèmes offrent un point de vue libre (free viewpoint video) sur la personne observée, permettant seulement une présence visuelle de l'utilisateur [47], aucune interaction n'est possible. Chaque utilisateur transmet donc aux autres utilisateurs une vue calculée de sa personne en fonction du point de vue virtuel des autres utilisateurs. Il faut donc calculer et envoyer autant de points de vue différents qu'il y a d'utilisateurs en train d'observer. Il faut également que les autres utilisateurs transmettent leur point de vue sur la scène virtuelle pour calculer l'image à leur envoyer.

Cette technique a été utilisée dernièrement par la chaîne Américaine *CNN* pour faire « apparaître virtuellement » une journaliste, située dans une autre ville, sur leur plateau de télévision. Cette dernière se trouvait dans un studio équipé de 35 caméras, ce qui permettait d'interpoler la vue que les spectateurs avaient d'elle malgré les mouvements de la caméra sur le plateau.

FIGURE 3.4 – À gauche le présentateur sur le plateau de CNN à New York et à droite la présentatrice, située réellement à Chicago dans un studio multi-caméra, mais « téléprésente » sur le même plateau.

D'autres systèmes calculent seulement un modèle 3D partiel basé sur une carte de profondeur [52, 34, 80] qui peut être envoyée sur le réseau indépendamment

du point de vue de l'utilisateur. Ceci n'assure par contre qu'une présence visuelle et mécanique limitée, la détection de collision est plus difficile, par exemple. Les systèmes multi-caméra permettent aussi de transmettre une information visuelle et géométrique complète sur la personne. La nature des données transmises dépend fortement de l'algorithme utilisé pour calculer le modèle. La transmission de nuages de points colorés [26, 38] est intéressante pour la présence visuelle mais pas pour la présence mécanique alors que la transmission de maillage [66] permet d'assurer une présence plus complète.

Le projet Tele-immersion 3D [38], issu d'une collaboration entre les universités de *Pennsylvanie (Upenn)* et de *Caroline du Nord (UNC)*, utilise un système d'acquisition multi-caméra permettant de construire un modèle 3D sous forme d'un nuage de points colorés, avec une méthode dérivée des techniques de stéréo-vision. Ce modèle peut être transmis sur un site distant pour des applications de téléprésence. Deux personnes distantes peuvent interagir autour de données scientifiques (voir figure 3.5), chacune présente dans l'espace virtuel grâce à son modèle 3D et peuvent voir le modèle de l'autre sur un écran. Les interactions sont par contre contraintes par l'utilisation d'un périphérique externe localisé, la nature des données 3D ne permet pas d'interactions directes.

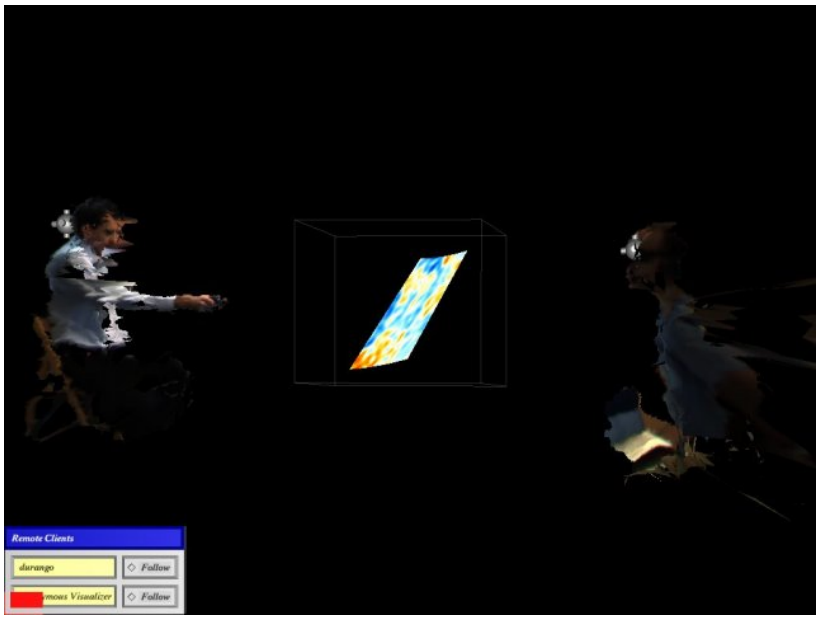
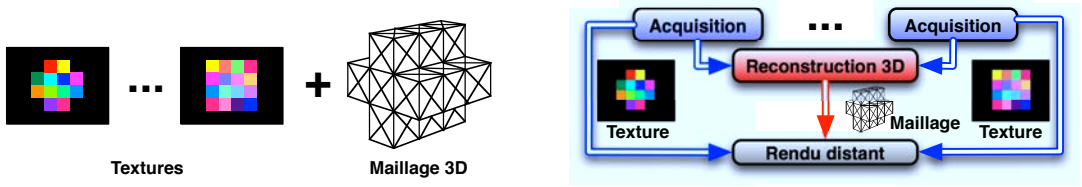


FIGURE 3.5 – Visualisation scientifique à distance et interaction (projet Tele-Immersion).

3.4 Implantation

3.4.1 Présence à distance et transfert de données

La visualisation à distance de modèles 3D générés à partir de notre système multi-caméra nécessite de transférer un maillage 3D, et ses textures associées (l'ensemble est appelé *méta-trame*, voir figure 3.6), avec les contraintes du réseau (bande passante limitée et latence minimum). Le maillage lui-même n'est pas grand consommateur de bande passante et peut-être facilement envoyé sur le réseau. Les textures, une par caméra, dans notre implémentation actuelle, requièrent une bande passante bien plus grande et représentent la majeure partie des transferts de donnée. Je présenterai une estimation de la taille des données à transférer dans la section 3.5 pour une configuration particulière.



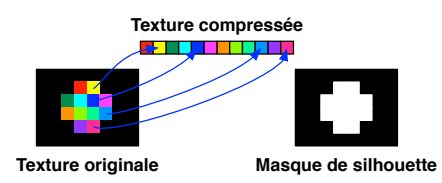
(a) Méta-trame (b) Schéma de transfert

FIGURE 3.6 – Transfert d'une méta-trame sans compression.

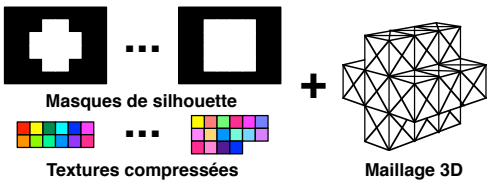
3.4.1.1 Compression des textures

Dans notre implémentation, les textures à transférer ne représentent que la partie à l'intérieur des silhouettes et non la totalité des images d'origine. Pour réduire les données à envoyer, les silhouettes sont utilisées comme masque afin de compresser et décompresser les images d'origine en une suite de pixels colorés à envoyer sur le réseau (voir figure 3.7-(a)). Il est donc nécessaire d'envoyer à tous les utilisateurs un modèle, un maillage, des textures compressées et les silhouettes binaires nécessaires à leurs décompressions (voir figure 3.7). Cette méthode permet de réduire les données au pire à 20% de leur taille d'origine, c'est en effet l'occupation maximum constatée des silhouettes dans les images pour une configuration standard et avec un seul utilisateur à modéliser.

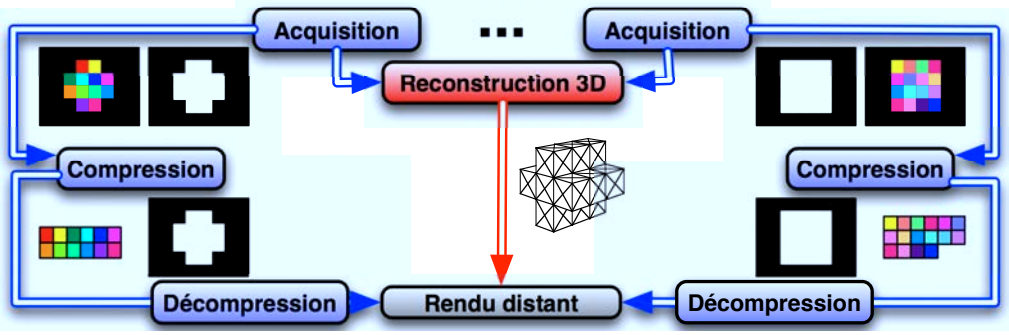
D'autres possibilités existent pour réduire encore plus le volume de donnée à envoyer, nous ne les avons malheureusement pas implémentées car ceci dépasse le champs d'application de nos travaux mais en voici une sélection :



(a) Principe de compression



(b) Méta-trame



(c) Schéma de transfert

FIGURE 3.7 – Transfert d'une méta-trame avec compression.

3.4.1.2 Sélection des textures

Il est, par exemple, possible de sélectionner les textures à envoyer en fonction du point de vue que les autres utilisateurs ont sur la scène. On pourra par exemple sélectionner, pour un utilisateur, seulement les textures pour lesquelles la direction de vue des caméras d'origine sont proches de la direction du regard de l'utilisateur concerné. Ceci nécessite par contre une communication supplémentaire, il faut en effet que l'utilisateur partage son point de vue sur la scène virtuelle afin que l'émetteur lui envoie un flux de textures différents de celui des autres utilisateurs (voir figure 3.8). Cette étape supplémentaire peut éventuellement faire perdre de l'interactivité à la visualisation, lors d'un changement de point de vue rapide sur la scène, les textures des modèles visualisés ne seront mises à jour qu'après communication de ce point de vue à l'émetteur et transfert des nouvelles textures associées au récepteur.

3.4.1.3 Création d'un atlas de texture

Une autre solution consiste à créer une texture unique, indépendante du point de vue des utilisateurs, appelée communément *atlas de texture*, elle peut être calculée à partir des images d'origine et du modèle 3D. L'image résultante est en moyenne à peine plus grande qu'une image d'origine, pour un seul utilisateur présent dans l'espace d'acquisition. Bien que cela apporte un gain au niveau de la

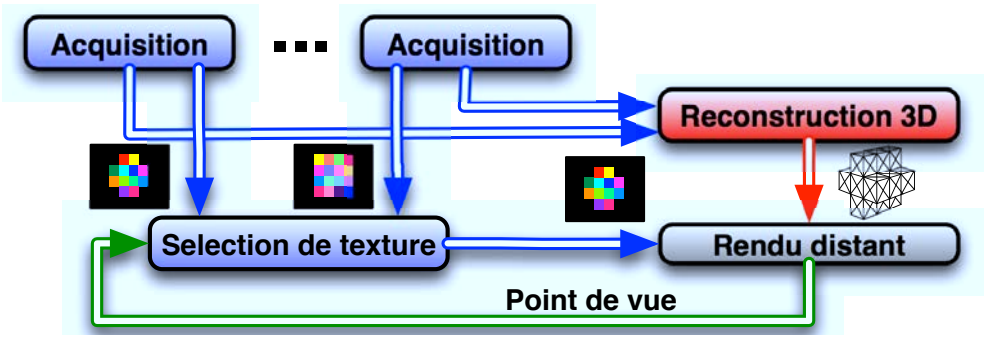


FIGURE 3.8 – Transfert avec sélection de texture.

charge réseau, cela implique une latence plus grande en contrepartie. En effet, il faut calculer cette texture avant envoi, étape qui ne peut être recouverte par une autre car le modèle 3D est nécessaire à sa réalisation. Il faut de plus, rassembler les données de textures sur un seul et unique noeud pour effectuer ce calcul avant envoi (voir figure 3.9). Avec la solution initiale, chaque noeud d'acquisition d'images peut envoyer directement ses textures compressées aux utilisateurs distants, étape qui peut d'ailleurs être recouverte par l'étape de reconstruction 3D.

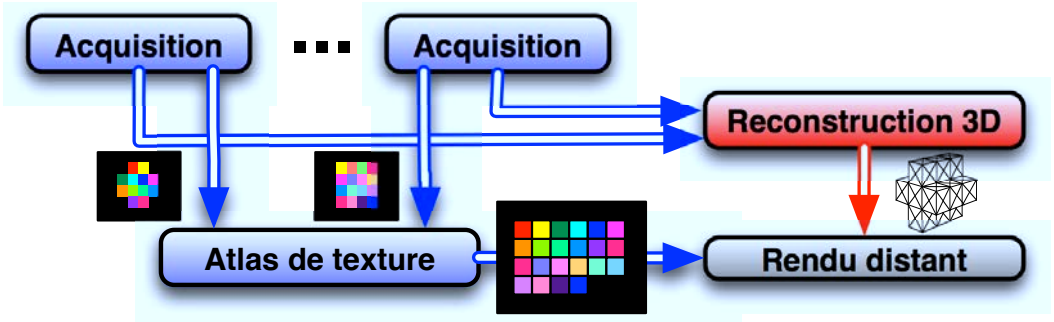


FIGURE 3.9 – Transfert avec création d'un atlas de texture.

3.4.1.4 Compression video classique

Il est également possible de compresser les textures de façon plus classique grâce aux méthodes de compression utilisée pour les flux vidéo. Le codec de compression *XVid* autorise la compression et décompression d'un flux vidéo en temps-réel. La compression utilise la cohérence temporelle supposée de deux trames consécutives. Suivant les réglages utilisés, le taux de compression peut être très intéressant. Nous avons testé cette solution mais elle comporte tout de même un désavantage, la machine qui effectue le rendu doit être capable de décompresser simultanément autant de flux qu'il y a de textures (voir figure 3.10). Ceci peut représenter un problème pour les performances du système et peut induire de la latence.

Une autre solution envisageable est l'utilisation de la compression de texture *DXC* de *NVIDIA*. Cette solution permet de compresser les textures sur le CPU de chaque noeud comportant une caméra, de l'envoyer aux noeuds se chargeant du rendu et de décompresser les flux de textures sur le GPU du noeud de rendu directement (voir figure 3.10). Cette compression ne s'effectue pas sans perte mais propose un bon rendement pour une altération visuelle très acceptable.

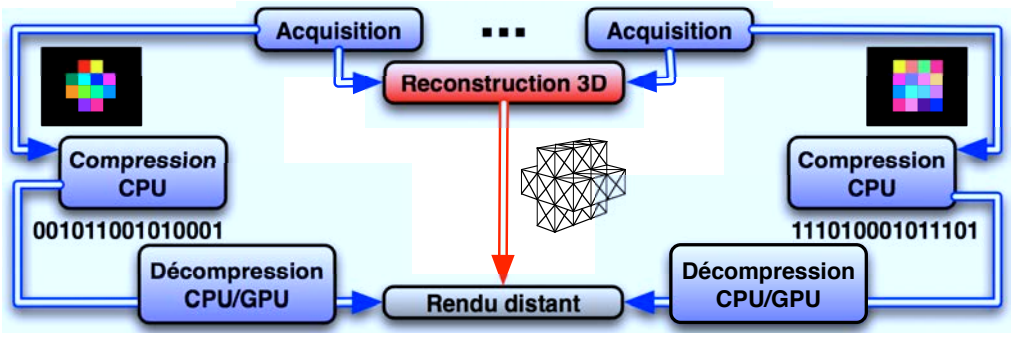


FIGURE 3.10 – Transfert avec compression/décompression classique de flux vidéo.

3.4.2 Synchronisation des données et ré-échantillonnage

Comme dans la version locale de l'application de reconstruction et visualisation 3D, les données géométriques doivent être synchronisées avec les données photométriques, c'est à dire, le flux de textures venant des noeuds d'acquisition doit être rendu au même instant que le maillage 3D venant du noeud de reconstruction 3D. Dans le cas contraire, cela peut entraîner des artefacts de visualisation. Cette synchronisation est gérée par l'intergiciel *FlowVR*.

L'étape de synchronisation évite également de saturer le réseau en ré-échantillonnant le flux - en supprimant des *méta-trames* (maillage et flux de textures) - dans le but de n'envoyer que des données à jour aux noeuds de visualisation distants (voir figure 3.11). Ce ré-échantillonnage s'effectue indépendamment pour chaque paire d'utilisateurs et permet donc de s'adapter à des réseaux ou des performances de visualisation différentes entre utilisateurs et de minimiser la latence en diminuant temporairement la fréquence d'envoi.

Comme la simulation physique ne nécessite, quant à elle, que des données géométriques pour calculer des collisions, chaque site envoie un flux de maillage au serveur gérant les objets dynamiques. Dans la mesure où le maillage n'est pas limitant pour la bande passante, et que la simulation physique peut s'exécuter à des fréquences supérieures à celles de la reconstruction 3D, les utilisateurs peuvent envoyer de nouveaux maillages dès qu'ils ont été calculés, sans étape de synchronisation ou de ré-échantillonnage.

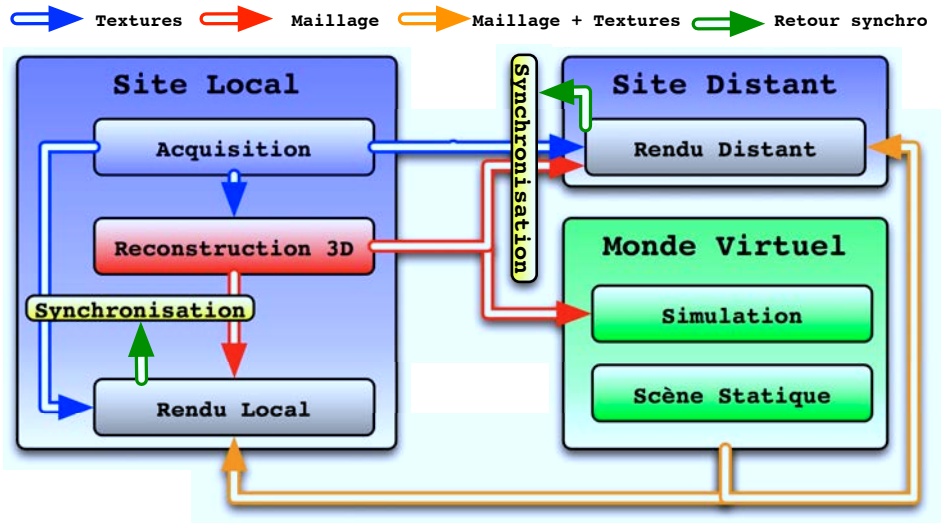


FIGURE 3.11 – Schéma de synchronisation simplifié.

3.5 Expérimentations

Afin de démontrer la faisabilité de l'utilisation des plateformes de reconstruction 3D multi-caméra dans un contexte de téléprésence et de collaboration à distance, nous avons mis en place deux expérimentations.

3.5.1 Simulation physique partagée

Notre premier démonstrateur a été réalisé afin de montrer que deux personnes, utilisant chacune une plateforme de reconstruction 3D, peuvent interagir en collaboration, et manipuler un objet virtuel dont le déplacement est régi par un moteur de simulation physique.

3.5.1.1 Configuration

Les deux plateformes d'acquisition sont des cubes d'1 mètre de côté équipés chacun avec 8 caméras firewire d'une résolution d'1 MPixels. Chaque caméra est branchée à un mini-PC utilisé pour l'acquisition et le traitement d'images. Un serveur embarquant un processeur bi-xeon sert pour les calculs de reconstruction 3D sur chaque plateforme. Ces plateformes sont des copies de celle présentée à *Siggraph* en 2007 [3]. Les deux plateformes sont connectées par un lien ethernet Gigabit au travers d'un PC qui sert de passerelle et de serveur pour la simulation physique.

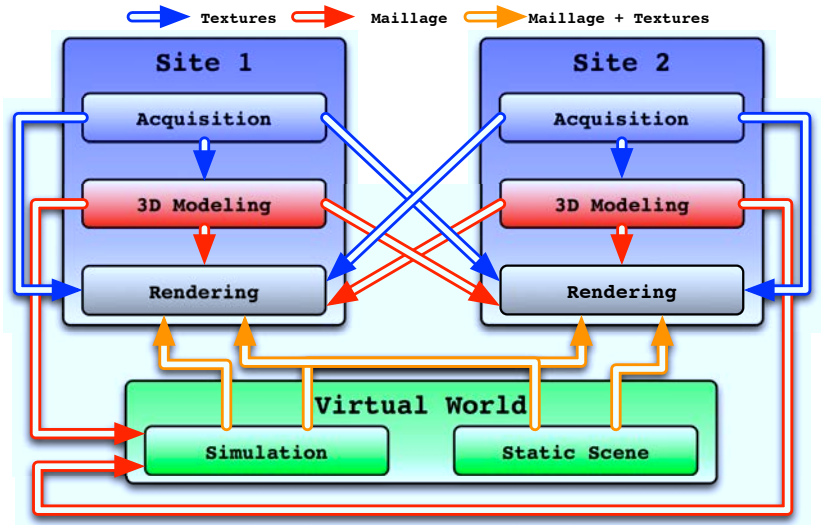


FIGURE 3.12 – Schéma de l'application de simulation physique partagée.

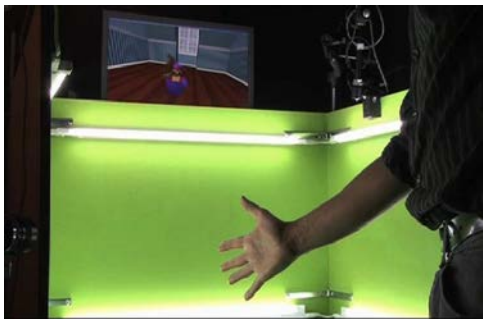
3.5.1.2 Estimation des données

Une plateforme d'acquisition produit des images de 1 MP, chaque image pèse donc 3 Mo et le flux d'images représente 24 Mo. Comme expliqué précédemment, la partie de l'image nécessaire pour texturer le modèle est celle à l'intérieur des silhouettes, ceci est utilisé pour réduire les données à transférer. Normalement, l'espace utilisé par les silhouettes est inférieur à 20% de l'image. Le flux de texture est donc réduit à 4.8 Mo. Le masque des silhouettes est également nécessaire au décodage des textures. Le flux de silhouette non compressé représente 1 Mo (il peut être négligeable après une compression RLE). Le flux total est donc d'environ 5.8 Mo. Afin de réduire la bande passante nécessaire nous avons décidé de diminuer la résolution des textures, avant envoi, à la moitié de leur résolution, ce qui réduit le flux à 1.45 Mo à chaque itération. Le maillage représente moins de 80 Ko (environ 10 000 triangles).

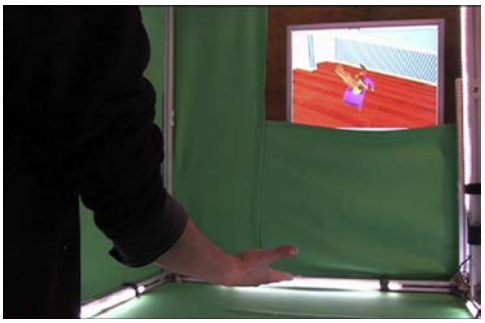
Si on considère une fréquence de 20 trames par seconde (noté fps pour *frame per second* en anglais) en moyenne, ce qui est raisonnable pour un affichage interactif, chaque plateforme nécessite une bande passante de 29 Mo/s pour envoyer son modèle et ses textures à l'autre. Ceci est bien dans l'échelle de grandeur d'un réseau ethernet Gigabit (120 Mo/s). Pour les interactions, chaque utilisateur peut envoyer directement, à la fréquence de calcul, son maillage 3D au serveur de simulation physique.

3.5.1.3 Résultats

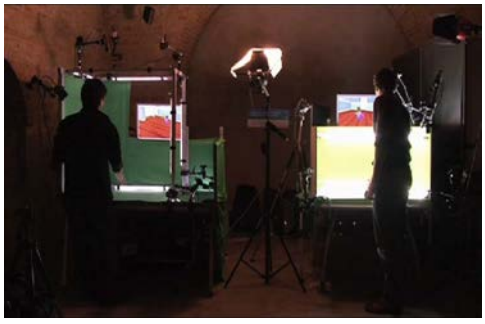
Les plateformes sont capables de produire des maillages 3D à une fréquence comprise entre 20 et 30 fps. La simulation physique et la visualisation tournent respectivement entre 50 et 60 fps et entre 50 et 100 fps, fonction de la charge du système. Comme ils s'exécutent de façon asynchrone à la génération de modèles 3D et textures, nous avons besoin de ré-échantillonner le flux de maillage et de textures indépendamment selon qu'il soit destiné au rendu local ou distant, ou à la simulation physique. En pratique, le transfert de maillage et de textures entre les plateformes oscille entre 15 et 20 fps, en fonction de la taille des silhouettes dans les images. Parallèlement, le transfert entre la reconstruction 3D et la visualisation sur une même plateforme et le transfert vers le serveur de simulation physique tournent toujours autour de la fréquence de production de maillage. Nous n'avons pas constaté de latence supplémentaire due au transfert du modèle 3D et des textures sur l'autre site. Pendant l'exécution, nous n'avons pas non plus saturé le lien ethernet Gigabit.



(a) Plateforme 1



(b) Plateforme 2



(c) Installation

(d) Monde Virtuel

FIGURE 3.13 – Démonstration à VRST 2008.

La précision du modèle obtenu en utilisant *EPVH* est satisfaisante aussi bien pour la visualisation que pour la simulation physique. Le niveau de détail obtenu avec ce type de plateforme permet de distinguer les différents doigts d'une main. Notre système résiste bien aux variations de charge réseau et processeur, les

données transférées peuvent être ré-échantillonnées pour éviter de trop grandes latences.

Ce démonstrateur a été présenté à la conférence *VRST* 2008 à Bordeaux [58] (voir figure 3.13 et la vidéo 1). Les deux plateformes étaient côte à côte, chaque utilisateur pouvait mettre sa main et l'apercevoir reconstruite en 3D dans une pièce virtuelle en compagnie de celle de l'utilisateur de la deuxième plateforme. Ils pouvaient ainsi interagir avec une marionnette virtuelle, animée par le moteur de simulation physique. Ils pouvaient, par exemple, essayer de collaborer pour attraper la marionnette en utilisant chacun qu'une seule main. Il était, bien sûr, impossible d'interagir physiquement entre utilisateurs. Les maillages s'intersectent simplement et la position des deux mains est superposée dans le monde virtuel.

Les personnes qui ont participé à l'expérimentation étaient satisfaites du système. Notre application manquait d'un but de collaboration précis pour les utilisateurs mais certains arrivaient à effectuer des actions en collaboration sans entraînement préalable. Les limitations pointées par les utilisateurs étaient plus liées au mode de visualisation et à la faible immersion (écran 2D placé devant la personne et vue à la troisième personne sur le monde virtuel), qu'au mode de collaboration.

3.5.2 Grille interactive

Notre second démonstrateur a pour but de démontrer la faisabilité de l'utilisation de la reconstruction 3D multi-caméra dans un environnement virtuel partagé permettant à des utilisateurs de se retrouver en 3D et d'interagir le plus naturellement possible. L'environnement est déployé sur une grille de calcul à l'échelle de la France afin que plusieurs utilisateurs puissent se connecter depuis des sites distants. De grands jeux de données sont également utilisés comme base de discussion et de collaboration entre les utilisateurs dans le monde virtuel.

3.5.2.1 Architecture

Dans la mesure où cette application est plus complexe en termes de déploiement que le premier démonstrateur nous avons fourni plus d'efforts pour la rendre le plus modulaire et efficace possible. Étant donnés le nombre et la variété des ressources mises en oeuvre : les machines, les caméras, les moniteurs, les réseaux locaux ou distants, etc. et l'hétérogénéité des processus de calcul impliqués, il est nécessaire de s'attarder sur la conception logicielle pour faciliter les tests, le déploiement et l'exécution de l'application finale. Nous utilisons pour cela l'intergiciel *Flow-*

VR (voir section §2.2.5) pour développer et tester chaque composant logiciel indépendamment et les assembler ensuite pour définir l'application.

Notre application combine 3 types de composants principaux, définis eux-mêmes par des sous-composants :

Composant de rendu : Un composant chargé du rendu est en charge de produire des images en fonction du type de périphérique de visualisation. Ce composant a deux ports d'entrée, un pour recevoir un flux de primitives 3D et un autre permettant de contrôler le point de vue de l'utilisateur sur la scène (voir figure 3.14). Les primitives 3D définissent les objets à inclure dans la scène. Une primitive 3D est auto-contenue, elle contient toutes les données (maillage, textures, positions) et le code des *shaders* qui définissent le rendu [4]. Les composants chargés de la visualisation ont donc la responsabilité de rendre les primitives qu'ils reçoivent. Les primitives peuvent être émises depuis plusieurs sources distribuées, c'est une des clefs de la flexibilité de l'application. De plus, ce composant a la possibilité, sur son port de sortie, d'émettre des données relatives au point de vue ou à la configuration actuelle du composant.

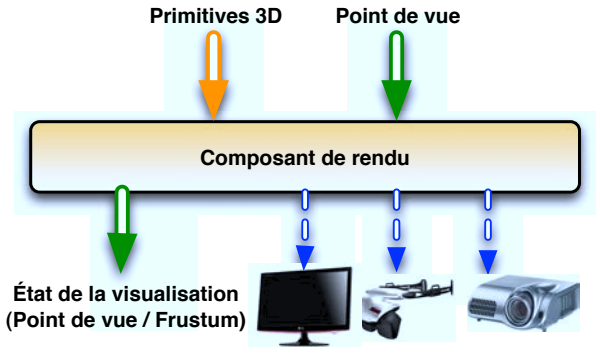


FIGURE 3.14 – Composant de rendu avec ses E/S.

Composant producteur de données graphiques : Les primitives 3D sont quant à elles produites par un autre type de composant, les producteurs de données graphiques. Ces composants reçoivent différent type d'événement et produisent des primitives graphiques qui seront envoyées à tous les composants de rendu (voir figure 3.15). Les plateformes de reconstruction 3D sont par exemple encapsulées dans ce type de composant, tout comme les serveurs de données dynamiques. De même, tout objet graphique est un producteur de données, qu'il soit statique ou dynamique. Selon le besoin, les producteurs de données envoient la même information à tous les composants de visualisation ou une information différente. Ils peuvent également nécessiter des informations des composants de visualisation pour adapter les données à envoyer, on verra que dans notre cas, le serveur de données a besoin de connaître le point de vue des utilisateurs à qui il envoie des données graphiques afin d'adapter et de réduire les transferts.

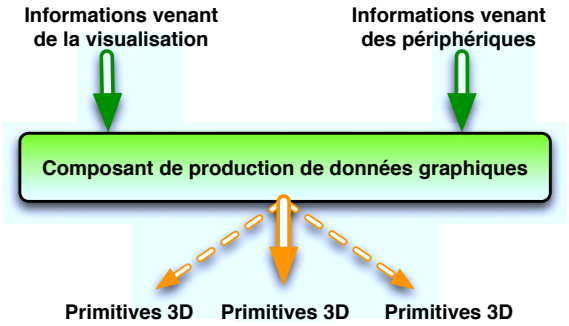


FIGURE 3.15 – Composant producteur de données graphiques avec ses E/S.

Composant de gestion des périphériques d'entrée : Un dernier type de composant est lié aux périphériques d'entrée. Ces composants envoient des événements qui sont transmis aux producteurs de données 3D, et/ou directement à la visualisation dans le cas où cet événement affecte le point de vue de l'utilisateur du périphérique (voir figure 3.16). Nous pouvons utiliser différents types de périphériques sous réserve de développer les sous-composants adaptés. Par exemple, nous nous servons d'une manette de jeux pour gérer les déplacements dans le monde virtuel, le composant associé produit des matrices de déplacements qui sont transmis au producteur de données local qui se charge de produire le modèle de l'utilisateur et transforme sa position en fonction de cette matrice. Ces matrices sont également envoyées au composant de rendu local afin de gérer le point de vue sur la scène en fonction de la position de l'avatar de l'utilisateur.

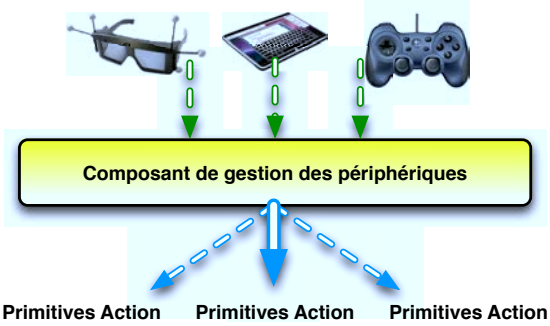


FIGURE 3.16 – Composant de gestion des périphériques avec ses E/S.

Définition des composants : Ajouter un nouveau moyen de visualisation, un nouveau périphérique, ou une nouvelle source de primitives graphiques se fait en remplissant les patrons de sous-composants accessibles aux développeurs et en configurant le composant final en accord avec les fonctionnalités souhaitées.

Dès que la grille de calcul est spécifiée et que le nombre et le type d'utilisateurs est connue, les composants nécessaires sont créés. Chacun des processus des

sous-composants est associé à une ressource et l'assemblage final des composants s'effectue en créant les connections qui relient les différents types de composants. À l'exécution, le déploiement de l'application et le transfert des données sont gérés par l'intergiciel de façon transparente pour le développeur. Notre démonstrateur comptabilise pas moins de 270 processus distribués sur 16 noeuds totalisant jusqu'à 72 coeurs et distribués physiquement sur 3 sites distants.

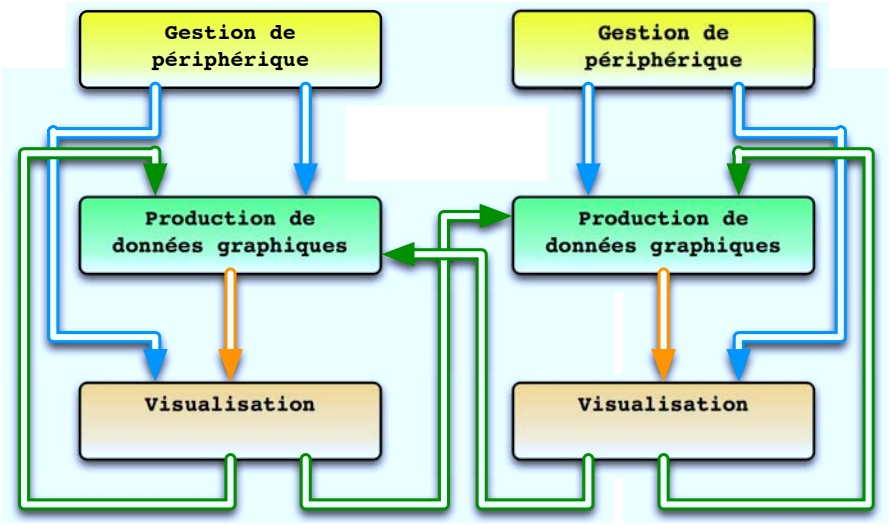


FIGURE 3.17 – Assemblage des composants entre eux.

3.5.2.2 Configuration

Notre démonstrateur type implique 3 sites : Grenoble, Bordeaux et Orléans, et un serveur de terrain. Chaque site spécifie ses composants types en fonction de ses moyens de visualisation, de ses périphériques et des données graphiques qu'il a à produire, à savoir l'avatar qui représentera l'utilisateur dans le monde virtuel. Un serveur de terrain est utilisé pour enrichir le monde et servir de base à des discussions et des interactions. Voici les spécificités de chacun des sites :

Orléans : La plateforme d'Orléans n'est pas équipée de système multi-caméra, elle se contente d'une seule caméra couleur. L'utilisateur filmé sera présent dans la scène virtuelle au travers d'une image affichée sur un *billboard* (technique utilisée pour plaquer une texture sur un quadrilatère dans une scène 3D) perpendiculairement au terrain et toujours en face des autres utilisateurs (voir figure 3.18-(b)).

L'utilisateur pourra déplacer sa représentation au moyen d'une manette munie d'un joystick qui lui servira également à déclencher des événements. Il visualise la scène virtuelle sur un grand écran placé devant lui, sur lequel est fixé la caméra (voir figure 3.18-(a)).

(a) Utilisateur (b) Représentation

FIGURE 3.18 – Configuration du site d'Orléans

Le site d'Orléans dispose donc d'un composant de type périphérique pour la manette, d'un composant de rendu pour gérer l'écran, et d'un producteur graphique qui gère la caméra et crée le *billboard*. Le débit nécessaire à l'envoi de l'image aux autres est négligeable car elle est compressée localement grâce au protocole *XVid* et décompressé sur les noeuds de rendu des autres utilisateurs.

Bordeaux : La plateforme de Bordeaux est équipée avec un système multi-caméra similaire à celui présenté dans la section § 2.2 composé de 8 caméras 0,5 MPixels et qui permet à un ou plusieurs utilisateurs d'être reconstruit en 3D. Le flux de maillage et de texture produit est d'environ 5 Mo par itération. Dans la mesure où le cluster et le réseau utilisés ne sont pas assez puissants pour faire s'exécuter la reconstruction 3D à la vitesse des caméras, la vitesse moyenne du système est de 16 fps. Le flux correspondant au maillage et à ses textures représente donc un débit de 80 Mo/s en moyenne.

La visualisation s'effectue sur un grand écran placé devant l'utilisateur et sur une tablette tactile qu'il tient dans ses mains. Cette tablette donne une vue du dessus du terrain à un niveau de détail réduit et permet des interactions 3D avancées grâce au paradigme d'interaction *Navidget* [27]. Avec une combinaison de gestes 2D, l'utilisateur peut se déplacer dans la scène ou déclencher des événements.

Le site de Bordeaux dispose donc d'un composant de périphérique d'entrée pour les actions réalisées grâce à la tablette, de deux composants de rendu, un pour l'écran de projection et un, allégé, pour la tablette, et d'un composant pour produire les données graphiques relatives au modèle de l'utilisateur.

Grenoble : La plateforme de Grenoble est également équipée d'un système multi-caméra, mais celui-ci utilise 8 caméras 1 MPixels et tourne aux alentours de 24 fps. Ce qui correspond localement à un débit de 240 Mo/s.

(a) Utilisateur

(b) Tablette

(c) Représentation

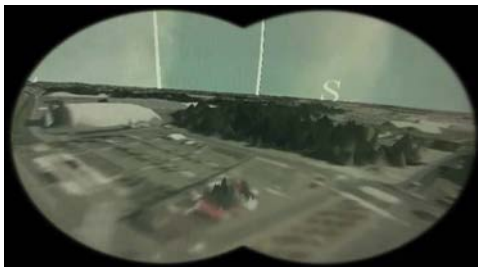
FIGURE 3.19 – Configuration du site de Bordeaux.

La visualisation s'effectue grâce à un casque de réalité virtuelle qui est suivi par un système optique, méthode d'immersion qui sera présentée dans la partie II. L'utilisateur peut bouger naturellement et se déplacer dans l'espace d'acquisition. S'il souhaite se déplacer plus loin dans le monde virtuel il peut utiliser une manette équipée d'un joystick. Manette qui lui servira également pour déclencher certains événements.

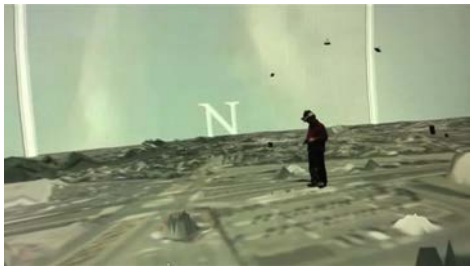
Le site de Grenoble dispose donc de deux composants de périphérique d'entrée, un pour le suivi du casque et un autre pour la manette, d'un composant pour le rendu dans le casque, et d'un composant pour produire les données graphiques relatives au modèle de l'utilisateur.

Serveur de terrain : Le serveur de terrain utilise des données réelles, prises par un scanner LIDAR monté sur un avion, avec une résolution d'un mètre. Le jeu de données consiste en de centaines de tuiles d'environ 2 millions de triangles. Pour un point de vue donné, il est usuellement nécessaire de n'afficher que 9 tuiles, soit environ 140 Mo de données, ce qui ne peut pas être transmis sur le réseau à un taux de rafraîchissement interactif. C'est pourquoi le serveur se repose sur un algorithme parallèle de niveau de détail afin de réduire le volume de données à transférer. Les données sont reconstruites en fonction de leur distance au point de vue, de leur intérêt et du niveau de détail désiré. Ceci permet d'adapter également le niveau de détail en fonction des capacités de chacun en terme de bande passante et d'affichage graphique. Le serveur de terrain consiste donc en un seul composant

(a) Utilisateur



(b) Visualisation

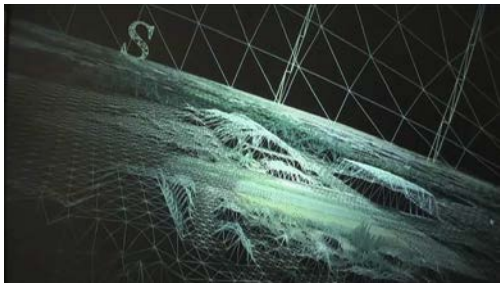


(c) Représentation

FIGURE 3.20 – Configuration du site de Grenoble.

de type producteur de données graphiques et nécessite d'avoir un retour des composants de visualisation pour connaître le point de vue à leur renvoyer.

Dans notre configuration, le serveur de terrain se situe à Grenoble. Une itération comportant des informations pour un nouveau point de vue et un niveau de détail maximum contient environ 80 Ko de données. À un taux de rafraîchissement de 50 fps, les besoins en bande passante sont au maximum de 96 Mo/s pour 3 points de vue et un point de vue supplémentaire, à un niveau de détail inférieur, pour la tablette de Bordeaux. Ceci est une valeur maximale atteinte seulement si les 3 utilisateurs bougent simultanément.



(a) LOD en fil de fer

(b) LOD texturé

FIGURE 3.21 – Terrain en niveau de détail (LOD) qui sert de base à l'environnement virtuel.

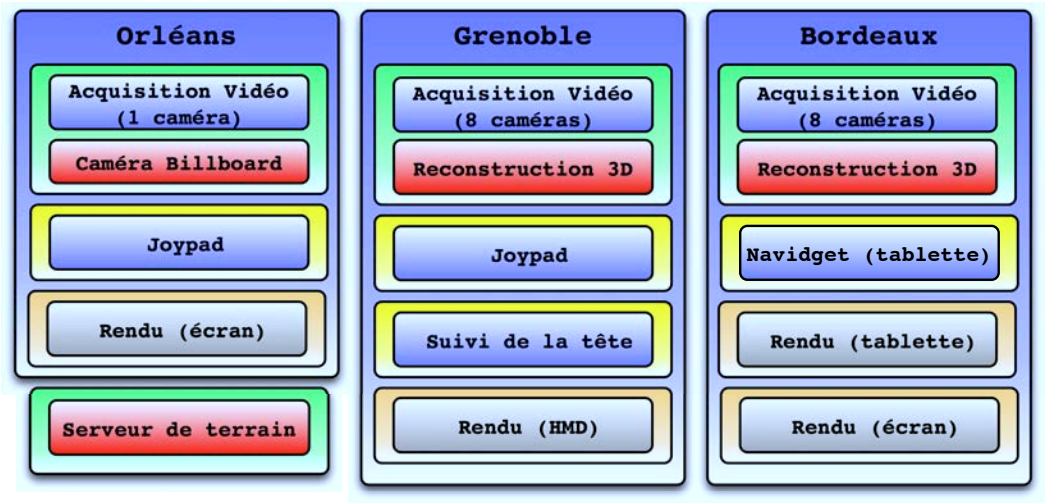


FIGURE 3.22 – Schéma général de notre configuration.

Réseau : Voici les configurations réseau de chacun des sites :

- Orléans utilise un réseau ethernet interne de 1 Gbits/s pour son cluster et une connexion de 100 Mbits/s à Internet.
- Bordeaux utilise également un réseau interne ethernet de 1 Gbits/s et une connexion de 1 Gbits/s à Internet.
- Grenoble utilise un réseau interne DDR Infiniband de 20Gbits/s et a une connexion de 10 Gbits/s à Internet.

Le trafic entre Bordeaux et Grenoble passe par un lien dédié de 10 Gbits/s grâce au réseau de la grille expérimentale *Grid'5000* alors qu'Orléans a seulement accès à l'Internet standard pour ses communications vers Bordeaux et Grenoble.

3.5.2.3 Résultats

Le tableau 3.1 montre le trafic moyen mesuré entre les sites pendant l'exécution. Le site de Grenoble bénéficie d'un réseau haute performance et de noeuds de calcul puissants, et peut donc traiter des quantités de données plus importantes que les autres sites. Ces derniers souffrent de congestion réseau et de surcharge des processeurs et ne peuvent recevoir la totalité des données, ainsi ils font appel aux mécanismes de ré-échantillonnage prévus pour transférer un flux, le plus à jour possible, en fonction des possibilités de chacun.

Nous avons également rassemblé des données de latence en mesurant le temps entre l'acquisition d'images et le rendu du modèle 3D correspondant sur un site distant. Par contre, cette mesure ne prend pas en compte le temps d'acquisition d'images par les caméras, celui du rendu sur le GPU et le temps d'affichage sur le moniteur. La figure 3.23 montre que les latences varient significativement entre

TABLE 3.1 – Trafic réseau moyen mesuré entre les 3 sites (en Mo/s) avec un utilisateur par site présent dans sa plateforme. Les termes diagonaux représentent le trafic total mesuré sur chaque site.

	Bordeaux	Grenoble	Orléans
Bordeaux	120	175	49
Grenoble	185	600	47
Orléans	2	2	2

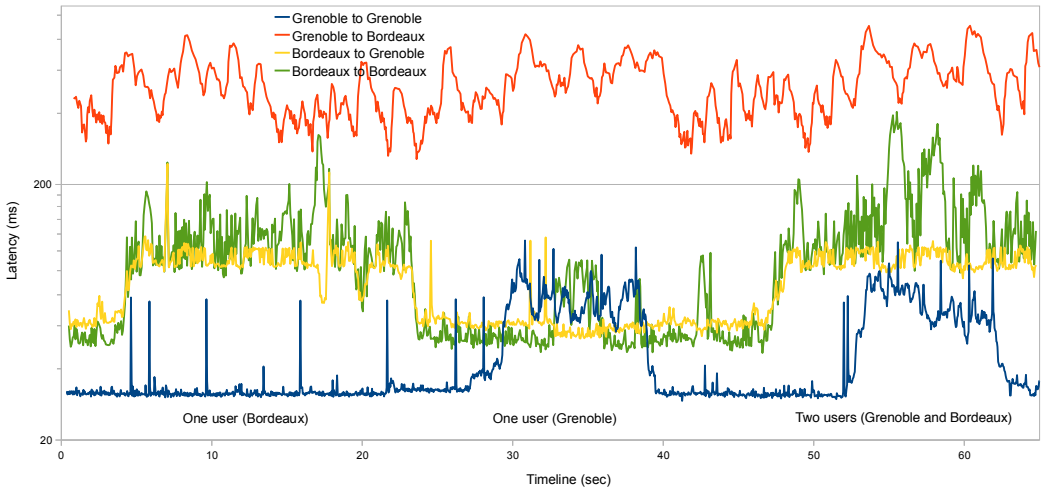


FIGURE 3.23 – Latences de la reconstruction 3D, à Bordeaux (8 caméras à 16 fps) et Grenoble (8 caméras à 24 fps), avec 1 ou 2 utilisateurs (échelle logarithmique sur l'axe Y). Les chiffres d'Orléans sont omis pour plus de clarté.

Grenoble et Bordeaux à cause des différentes capacités des clusters et des bandes passantes des réseaux.

À Grenoble la latence reste en dessous de 100 ms pour le modèle 3D local et aux alentours de 100 ms pour le modèle reçu depuis Bordeaux. Ces valeurs sont acceptables pour des interactions. À Bordeaux, la latence pour le modèle 3D local est d'environ 140 ms à cause de ses capacités moindres en termes de réseau et de puissance de calcul. Le modèle 3D transmis entre Grenoble et Bordeaux mène à une latence importante de l'ordre de 500 ms. Nous ne savons toujours pas à quoi est dû cette asymétrie dans les performances réseau entre Grenoble et Bordeaux, il est possible que la cause soit d'ordre matériel. Comme prévu, les latences à Orléans sont très importantes également (plus de 500 ms) du fait des capacités réseau limitées du site.

Ces tests nous ont montré l'importance d'une bonne bande passante et d'une faible latence sur les résultats obtenus. Tout comme les débuts de la vidéo-conférence par internet, il est important d'avoir une bonne connexion pour profiter pleinement du système. Les barrières de synchronisation et de ré-échantillonnage

permettent de ne pas accumuler de latence en réduisant temporairement la fréquence si la connexion est saturée. Il y a également un compromis à faire entre la compression des données qui peuvent ajouter de la latence de calcul et l'envoi de données brutes qui peuvent saturer le lien réseau et ajouter de la latence de transfert.

3.5.2.4 Application

Mise en place : Afin de démontrer le potentiel de notre grille interactive et de la tester dans un cadre applicatif, nous avons crée un scénario de collaboration à distance basé sur le démonstrateur décrit précédemment. En gardant la même configuration, nous proposons aux trois utilisateurs d'évoluer librement sur le terrain, une modélisation d'une partie de la ville d'Orléans, et de discuter d'un projet d'aménagement ensemble. Ils peuvent interagir, grâce à leurs périphériques d'entrée, et ajouter des infrastructures sur le terrain. Ils utilisent également un système de télé-conférence classique pour pouvoir se parler.

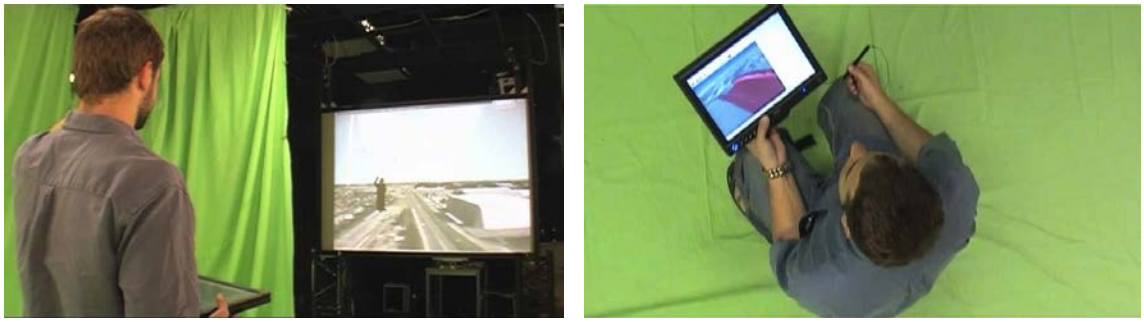
Cette application a été présentée dans une publication accompagnée d'une vidéo lors de la conférence *ACM Multimedia* 2010 à Florence en Italie [56] (voir la figure 3.24 et la vidéo 2).

Discussion : Globalement les utilisateurs sont très satisfaits par la sensation de présence et les capacités de discussion offertes par ce genre de plateforme. La présence visuelle offerte par la reconstruction 3D complète la conversation audio des utilisateurs. Ils peuvent en effet dialoguer en utilisant des gestes pour montrer des positions sur la carte, leur langage corporel est aussi véhiculé. Les meilleures conditions ont été expérimentées à Grenoble, où la fréquence était maximale et la latence minimale.

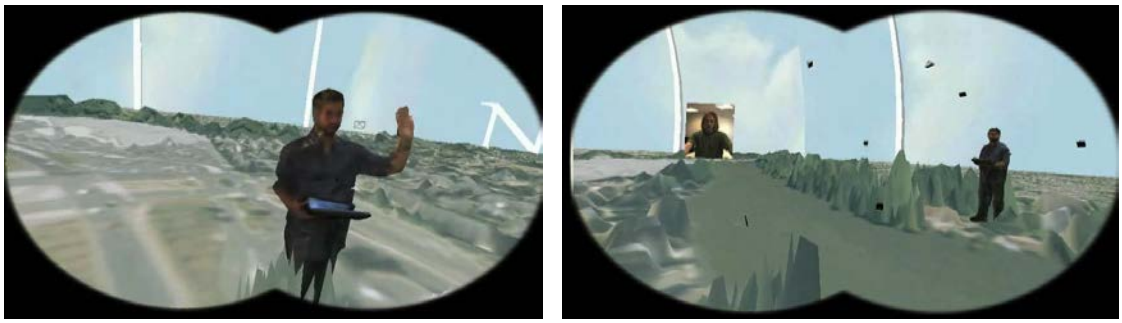
D'un point de vue plus technique, cela nous a permis de voir les limitations de notre système de déploiement d'application. En effet, nous utilisons *FlowVR* pour déployer les processus qui tournent sur chacun des clusters. La connexion au monde virtuel n'est donc pas dynamique, un lancement statique de toute l'application doit être effectué depuis l'un des sites. Si l'un des sites a un problème logiciel ou matériel, il ne peut pas se déconnecter et reconnecter sans relancer toute l'application. De même si un problème apparaît au lancement de l'application il faut la relancer du début ce qui fait perdre beaucoup de temps surtout en phase de test. Il est donc également impossible de se connecter à une application déjà lancée car le réseau de connexion entre les utilisateurs est déjà déterminé avant le lancement. Ce problème est néanmoins en passe d'être réglé : Une des prochaines versions de *FlowVR* devrait permettre un lancement de l'application par partie, ce

qui rendrait également possible de redémarrer un morceau de l'application globale sans tout redémarrer.

(a) Site d'Orléans



(b) Site de Bordeaux



(c) Site de Grenoble

(d) Environnement Virtuel

FIGURE 3.24 – Application vue depuis les différents sites.

3.6 Bilan

Nous avons donc, grâce à ces deux expériences, montré la faisabilité d'utilisation des systèmes de reconstruction 3D multi-caméra dans un cadre de téléprésence et de collaboration à distance. Les avantages de ces systèmes sont liés au degré de présence dans les mondes virtuels permis par la modélisation 3D temps-réel. Le fait que notre avatar soit exactement à notre image et que son déplacement et ses expressions soient directement liées aux nôtres enrichissent les collaborations entre les utilisateurs.

Les performances globales en terme de fréquence et de latence pourraient être encore améliorées en se concentrant sur la compression des données envoyées à distance. L'amélioration de la réactivité de la reconstruction 3D aiderait à réduire la latence entre utilisateurs et pourrait permettre d'effectuer des tests sur des réseaux qui offrent une bande passante moins importante.

Les interactions basées sur la détection de collisions entre notre modèle 3D et des objets virtuels sont également possibles dans un contexte multi-utilisateur et en collaboration à distance. De nouveaux algorithmes distribués doivent donc être développés dans un soucis de passage à l'échelle du nombre d'utilisateurs.

D'autres métaphores d'interaction doivent être développées pour venir enrichir les possibilités de collaboration. Le domaine émergeant de la reconnaissance de gestes et d'actions devrait pouvoir apporter une solution à ces limitations en matière d'interaction.

L'interaction physique entre utilisateurs manque également dans de tel système. Il n'existe encore aucun système de retour d'effort permettant de produire une force de façon dense sur toute la surface du corps de l'utilisateur, et encore moins de système non-invasif.

II

Immersion et interaction

In his essay “Of Other Spaces” (1986) Foucault describes the mirror image as a paradoxical amalgam of the real and the virtual where the self is both present and absent : « In the mirror, I see myself there where I am not, in an unreal, virtual space that opens up behind the surface ; I am over there, there where I am not, a sort of shadow that gives my own visibility to myself, that enables me to see myself there where I am absent... »
Foucault 1986

Chapitre 4

Interaction à la première personne

4.1 Motivations et problématiques

La première version de la plateforme *GrImage* présentée au chapitre §2.2 permet donc aux utilisateurs présents dans l'espace d'acquisition d'être modélisés en 3D en temps-réel. Cette modélisation, enrichie par les textures issues des images d'origine, peut servir de représentation visuelle au sein d'un environnement virtuel. Elle peut également servir de donnée d'entrée pour un logiciel de simulation physique capable de détecter des collisions entre le modèle produit et des objets virtuels, et donc de base pour de nouvelles métaphores d'interaction. Dans cette version du système, l'utilisateur perçoit l'environnement virtuel sur un écran placé devant lui. Il peut donc voir son modèle 3D reconstruit évoluer en même temps que lui dans l'univers virtuel. C'est son *clone numérique*.

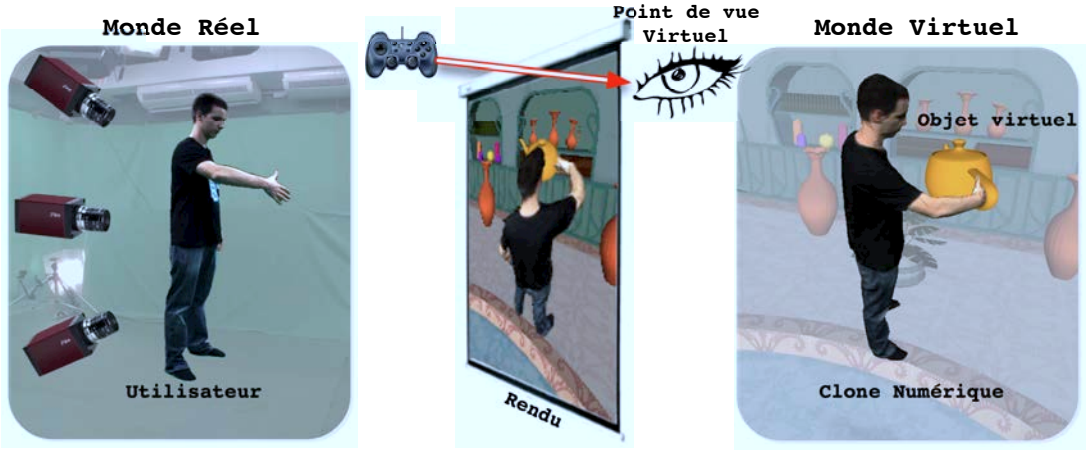


FIGURE 4.1 – Configuration de notre système avec visualisation et interaction à la troisième personne.

L'utilisateur a donc un point de vue et des possibilités d'interaction à la troisième personne : il effectue des actions à un endroit et il regarde son avatar

les réaliser par le biais de l'écran (voir figure 4.1). C'est le même type de point de vue et d'interactions que dans les jeux multi-joueurs en ligne, à la différence que les actions réalisables ne sont pas pré-programmées et qu'il n'est pas nécessaire d'utiliser un périphérique pour contrôler son avatar. Les interactions sont plus naturelles car le clone numérique effectue exactement les mêmes actions que l'utilisateur. Il n'y a pas besoin d'apprendre à se servir d'un tel système, les métaphores d'interaction sont très proches de celles que l'on expérimente dans la réalité. Par contre, comme expliqué précédemment, dans la mesure où le système ne détecte que des collisions, il n'est pas possible d'effectuer toutes les actions réalisables naturellement. La préhension est, par exemple, difficile car il n'est pas possible de récupérer des informations de force, de friction ou mêmes les concavités du modèle 3D.

Il nous est apparu, lors de nos expériences et démonstrations grand public, que l'utilisation de notre système n'était pas triviale pour tout le monde. En effet, il est nécessaire d'utiliser un périphérique physique pour piloter le point de vue que l'on veut avoir sur l'environnement virtuel 3D. Ce point de vue peut être très différent du point de vue que l'on a sur la scène réelle et donc être source de confusion et de désorientation pour les personnes qui n'ont pas l'habitude de naviguer dans des mondes virtuels. Il est par ailleurs connu que la visualisation à la première personne apporte une sensation d'immersion plus forte qu'une visualisation à la troisième personne.

L'utilisateur doit en plus interagir en regardant ce qu'il fait à l'écran. D'une part, il a l'impression de manipuler en aveugle les objets, ce qui peut demander un peu d'entrainement, d'autre part, cela limite l'espace d'interaction aux endroits où il voit correctement l'écran, ce qui est dommage étant donné que le système est capable de calculer un modèle 3D et des collisions dans tout l'espace d'acquisition.

Ces problèmes limitent le sentiment de présence de l'utilisateur dans le monde virtuel. Dans la mesure où il n'est pas complètement immergé, il n'a pas vraiment l'impression de ne faire qu'un avec son clone numérique.

Ces limitations nous ont poussées à chercher une solution qui permette à l'utilisateur d'interagir dans tout l'espace d'acquisition sans avoir à regarder un écran dans une direction et qui ne nécessite pas l'utilisation d'un périphérique pour contrôler son point de vue sur la scène virtuelle.

4.2 Conception

L'idée est donc de coupler un système de visualisation unidirectionnelle à la première personne avec notre système multi-caméra afin de permettre à l'utilisateur

d'effectuer des interactions à la première personne dans tout l'espace d'acquisition et de se sentir plus immergé dans le monde virtuel.

4.2.1 Système de visualisation à la première personne

Dans la suite de cette section, je présente un état de l'art des systèmes de visualisation unidirectionnelle à la première personne les plus courants, à savoir les *CAVEs* et les *HMDs* et je discute de leur possible intégration dans un système multi-caméra et des avantages et inconvénients de chacune des solutions.

4.2.1.1 Les CAVEs

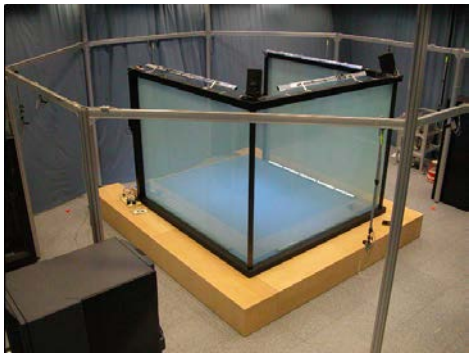
Les *CAVEs*, présentés succinctement dans la section §2.1.1.2 sont très utilisés pour la visualisation immersive à la première personne. Mais les environnements basés sur une projection à distance présentent un problème intrinsèque pour l'interaction : bien qu'il soit possible d'afficher le monde selon le point de vue de l'utilisateur, il est impossible d'afficher un objet virtuel dans la main de l'utilisateur. En effet, dans cette configuration sa main bloquera sa vision de l'écran et donc l'affichage de l'objet.

Les *CAVEs* présentent également l'inconvénient d'être limités à un seul utilisateur. En effet, le point de vue d'une seule personne peut être généré. Il existe une version utilisant un rendu actif et passif conjointement qui autorise à deux utilisateurs d'avoir un point de vue différent. Le problème d'occultation est encore plus important dans un contexte multi-utilisateur, il est impossible de placer des objets virtuels entre les utilisateurs, chacun d'eux bloquant une partie de la surface de projection.

Avec *Blue-C*, Gross et al. [26] ont démontrés la possibilité de faire fonctionner un système multi-caméra dans un environnement immersif de type *CAVE* (voir figure 4.2-(a)). Leur système multi-caméra permet d'obtenir une représentation de l'utilisateur sous forme d'un nuage de points colorés. Ce mode de reconstruction 3D permet d'afficher le clone numérique suivant n'importe quel point de vue virtuel (voir figure 4.2-(b)). Il n'est cependant pas facile de s'en servir pour des applications interactives, en effet aucun maillage n'est fourni ce qui limite, entre autres, le calcul d'interaction. Des algorithmes permettent de calculer un maillage à partir d'un nuage de points, mais ils ajoutent de la latence au système au dépend de l'interactivité .

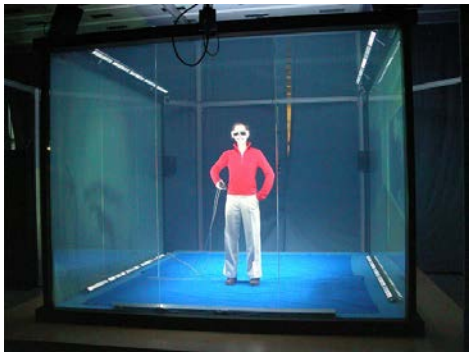
Le principal inconvénient de l'utilisation de caméras dans un *CAVE* réside dans le fait que le fond n'est pas statique. En effet, les projecteurs émettent de la

lumière sur les faces du cube, qui, en plus d'éblouir les caméras, empêchent toute soustraction de fond. Grâce à l'utilisation d'un système d'écrans actifs *Blue-C* résout le problème. Les faces du *CAVE* peuvent passer d'un état opaque (voir figure 4.2-(d)) pour la projection à un état transparent (voir figure 4.2-(c)) pour l'acquisition d'images. Les caméras peuvent donc « voir » à travers les façades du *CAVE*. Un système de synchronisation s'occupe de déclencher les caméras, les projecteurs stéréoscopiques, un flash de LEDs, et le contrôleur des écrans au moment opportun.

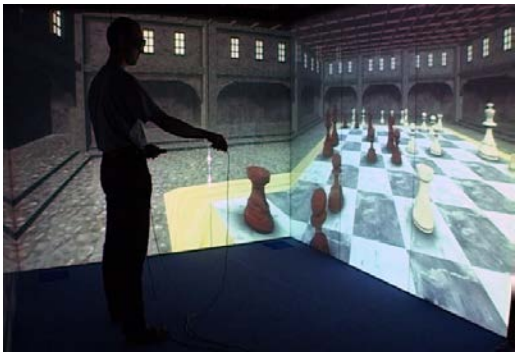


(a) Système

(b) Application



(c) Écran transparents



(d) Écran opaques

FIGURE 4.2 – Le système Blue-C en action.

Le sujet est donc immergé dans le monde virtuel, son point de vue est contrôlé grâce au suivi de sa tête et il peut profiter d'une vision en relief. Il peut également être présent visuellement dans un monde virtuel, par contre sa présence mécanique n'est pas assurée avec cette méthode de reconstruction 3D, il doit utiliser un périphérique pour interagir.

Cette solution répond donc partiellement à notre problématique mais présente le gros inconvénient d'être très onéreuse et techniquement difficile à mettre en oeuvre. Il serait possible d'utiliser notre système dans ce type d'environnement mais les limitations intrinsèques liées aux occultations ne permettraient pas de profiter pleinement d'un système d'interaction à la première personne. Seule la une vision unidirectionnelle et le sentiment d'immersion seraient intéressants.

4.2.1.2 Les HMDs

Une autre solution pour avoir un point de vue à la première personne de façon unidirectionnelle est d'utiliser un casque de réalité virtuelle (appelé *HMD* dans le reste du chapitre), que j'ai introduit dans la section §2.1.1.2. Ces derniers se composent d'une paire d'écrans placés devant les yeux de l'utilisateur et fixés à un casque ou à des lunettes. Les écrans peuvent être semi-transparents pour les applications de réalité augmentée (*optical see-through*) ce qui permet de voir l'environnement réel à travers les verres et de « l'augmenter » avec des objets virtuels affichés sur les écrans.

La taille et la résolution des écrans peuvent varier. Ils sont un facteur clef du sentiment d'immersion de l'utilisateur. Le poids et le confort sont très importants également car il faut que l'utilisateur puisse se promener le plus librement possible dans l'espace et qu'idéalement il fasse abstraction du casque et se sente vraiment présent dans l'univers qu'il observe. Les câbles d'alimentation et d'affichage auxquels il faut faire attention constituent une grosse limitation par rapport à de simples lunettes.

La plupart des casques proposent un affichage en relief en envoyant une image différente aux deux écrans. Leur intégration à un système existant comme le notre est assez simple, il suffit de brancher le casque à la place d'un moniteur et d'ajuster les paramètres d'affichage pour s'en servir.

Les casques ont également l'avantage de pouvoir passer facilement à l'échelle au niveau du nombre d'utilisateurs. En effet, sous réserve de pouvoir suivre simultanément plusieurs têtes d'utilisateurs et de pouvoir afficher les points de vue correspondants, il n'y a pas de problèmes majeurs à équiper d'un *HMD* plusieurs personnes présentes dans le même espace.

Notre préférence est donc allée à l'utilisation d'un *HMD* au sein de notre plateforme. Ses avantages en terme de possibilités d'interaction et sa facilité d'intégration à notre système ont été des facteurs prépondérants malgré le fait que la qualité visuelle soit moins satisfaisante.

4.2.2 Suivi du regard

Le système de visualisation unidirectionnelle permet à l'utilisateur d'avoir un rendu de la scène dans n'importe quelle position et orientation. Indépendamment de cela, il faut assurer à l'utilisateur d'avoir un point de vue à la première personne. C'est-à-dire que la caméra virtuelle qui assure le rendu du point de vue de la scène soit bien positionnée au niveau du regard de l'avatar. Ce mode de visualisation

est couramment utilisé dans les jeux de type *FPS* (*First Person Shooter*). Il est, dans le cas de ces jeux, évident de calculer la position du regard de l'avatar car son déplacement est régi par un périphérique de jeux.

Dans notre cas, il est plus compliqué de connaître le point de vue de l'avatar car il est directement lié au point de vue de l'utilisateur. Il faut donc détecter son regard à chaque fois que le noeud de rendu rafraîchit l'affichage. La solution la plus pratique est le suivi d'une cible attachée à la tête de l'utilisateur. Cette cible sera détectée et suivie dans le temps grâce à l'une des méthodes ponctuelles présentées dans la section 2.1.1.3. De nombreux systèmes commercialisés permettent de réaliser cette tâche, ce sont des traqueurs (voir figure 4.3-(b)). Nous nous servons, dans notre cas, d'un traqueur optique développé en interne, le *Cyclope* (voir figure 4.3-(a)) qui présente l'avantage d'être plus ouvert et donc de pouvoir s'intégrer plus facilement dans notre système en termes de synchronisation de données et de calibrage. Il existe d'autres types de traqueur, comme ceux présentés dans la section §2.1.1.3, mais les traqueurs optiques sont les plus précis et les moins contraignant.

(a) Le Cyclope et une cible

(b) Le TrackIR et sa cible

FIGURE 4.3 – Deux traqueurs 3D optiques basés sur une seule caméra et leurs cibles.

Il s'agit d'une caméra avec un capteur noir et blanc pourvu d'un filtre ne laissant passer que le rayonnement infrarouge, et équipée d'un flash constitué de LEDs émettant dans l'infrarouge. Un système autorise le déclenchement du flash uniquement pendant le période d'acquisition d'images. Les rayons infrarouges produits par le flash vont se refléter sur une cible, constituée de quatre sphères réfléchissantes non coplanaires, les rayons réfléchis sont ensuite captés par la caméra. Après seuillage de l'image, le centre des sphères est facilement détecté. L'algorithme *POSIT* [17] utilise ensuite la position relative des sphères en 3D et leur projection en 2D pour déterminer la transformation 3D de la cible par rapport au repère de la caméra infra-rouge (M_{tc}).

Cette solution ne donne pas exactement la position et l'orientation du regard de l'utilisateur mais celui de la cible fixée rigidement sur le support du *HMD*

(voir figure 4.4). Il faudra donc appliquer une transformation supplémentaire afin d'avoir la matrice de transformation du regard du sujet par rapport au repère du *Cyclope* (M_{ec}) avec la relation :

$$M_{ec} = M_{tc}.M_{et} \ . \tag{4.1}$$

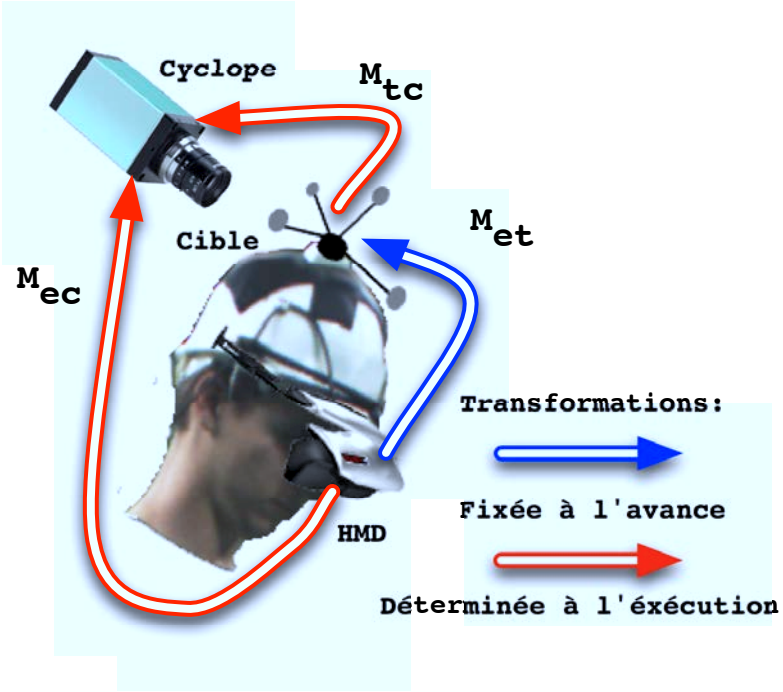


FIGURE 4.4 – Principe du suivi optique du regard de l'utilisateur.

La transformation à appliquer ne doit pas changer dans le temps, le *HMD* est fixe sur la tête de l'utilisateur et la cible rigidement liée au *HMD*, la transformation qui va du repère des yeux de l'utilisateur au repère de la cible (M_{et}) reste inchangée. Elle peut être calculée lors de la conception du casque ou calibrée [24]. Afin d'afficher une image en adéquation avec les propriétés des écrans utilisés, il est préférable de calibrer également les paramètres internes de la caméra virtuelle utilisée (largeur du champs de vision ou distorsion optique) pour rendre la scène le plus fidèlement possible.

Le calibrage de ces paramètres ne faisant pas partie de notre problématique initiale, nous avons opté pour un réglage manuel de la transformation et de l'angle du champs de vision. J'expliquerai plus en détails dans la section §4.5 comment le réglage de ces paramètre est effectué et dans quelles mesures son imprécision peut affecter l'expérience visuelle de l'utilisateur.

Finalement, si on définit la relation entre la position du *Cyclope* et le repère qui définit l'origine du monde virtuel (M_{cw}), on peut calculer à l'exécution la transformation du regard de l'utilisateur dans le monde virtuel (M_{ew}) avec :

$$M_{ew} = M_{cw}.M_{ec} \ . \tag{4.2}$$

On obtient ainsi un système de visualisation unidirectionnelle à la première personne. Il permet donc à un sujet de s'immerger et d'explorer un monde virtuel en se déplaçant librement dans les limites de l'espace où la cible qu'il a sur la tête est visible par le *Cyclope*.

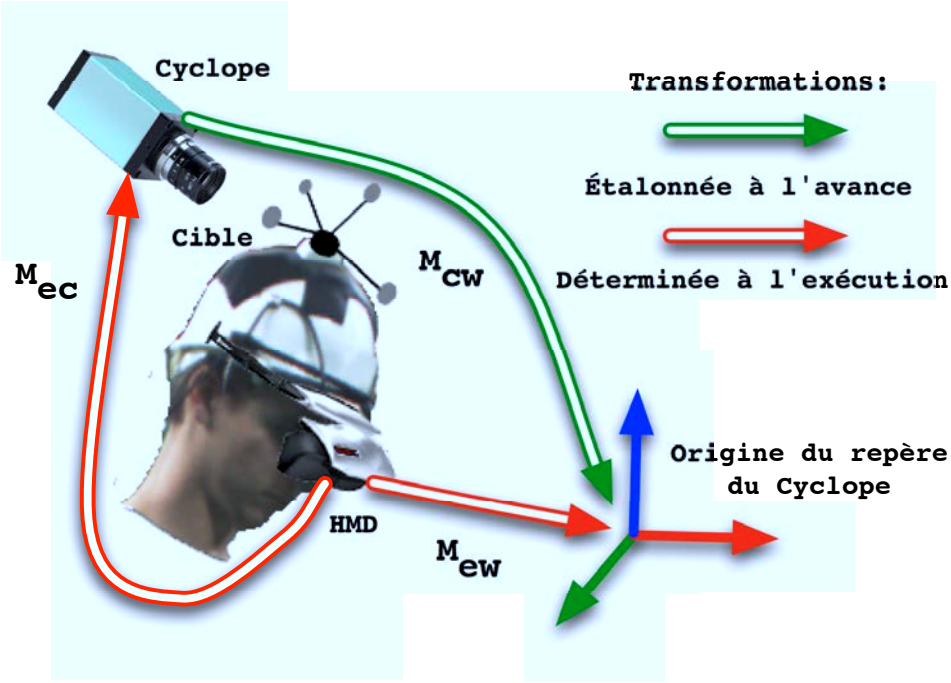


FIGURE 4.5 – Calibration de notre système de suivi optique.

Il existe d'autres solutions que les traqueurs 3D pour déterminer la position et l'orientation du regard de l'utilisateur. Par exemple, des algorithmes de traitement d'images capables de détecter la tête et les yeux de l'utilisateur. Ils peuvent par triangulation donner une position approximative des yeux dans l'espace 3D. Il est par contre plus difficile de déterminer l'orientation du point de vue de l'utilisateur avec cette méthode. Ces algorithmes manquent de toute façon de précision et de stabilité ce qui peut nuire au confort visuel de l'utilisateur. De plus, ils sont incompatibles avec le port de lunettes et encore plus avec un *HMD*.

Les *HMDs* sont couramment équipés de centrale d'inertie et d'accéléromètres afin de connaître l'orientation de la tête de l'utilisateur. Cette solution, bien que tout à fait suffisante pour déplacer le point de vue de l'utilisateur dans un jeux à

la première personne sur ordinateur, n'est pas adaptée pour l'exploration d'un monde virtuel où le sujet peut évoluer librement. En effet, les centrales d'inertie sont calibrées par rapport à une orientation initiale et l'information récupérée l'est relativement à cette orientation, elle tend à dériver dans le temps et donc à devenir imprécise. L'information de déplacement donnée par les accéléromètres est, quant à elle, très imprécise et tout à fait inutile pour calculer une position précise dans l'espace.

L'utilisation d'un traqueur optique s'est avéré être la solution la plus efficace et la plus précise pour suivre le point de vue de l'utilisateur et permettre une visualisation unidirectionnelle à la première personne. De plus, l'utilisation du *Cyclope* nous a permis de calibrer facilement le système dans le repère de l'espace d'acquisition.

4.2.3 Alignement des repères

À ce stade, nous avons un système capable de reconstruire en 3D l'utilisateur afin qu'il soit représenté le plus fidèlement possible dans un univers virtuel, et un système qui lui permet de percevoir cet univers virtuel à la première personne comme si il s'y trouvait vraiment. La fusion de ces deux systèmes est facilitée par leur indépendance, mais sans alignement, rien ne lie le point de vue de l'utilisateur à celui de son clone virtuel. En effet, la position de la caméra virtuelle est liée à la position définie entre le *Cyclope* et le repère d'origine du monde virtuel (M_{cw}), et la position de l'avatar est quant à elle liée à la relation entre le repère de l'espace d'acquisition, défini lors de l'étallonage des caméras, et le repère du monde virtuel (M_{vw}). Les déplacements de la tête de l'avatar et ceux du point de vue seront bien identiques, mais rien ne garantit que leurs positions coïncideront.

Afin d'aligner ces positions, nous tirons parti du fait que la caméra qui suit la tête de l'utilisateur est relativement semblable à celles utilisées pour le reconstruction 3D. Nous pouvons donc l'intégrer et la synchroniser aux autres caméras lors de la phase de calibrage du système. Avec un bâton sur lequel sont fixées des lumières émettant dans l'infrarouge, la détection de la baguette ne pose pas plus de problème que pour les autres caméras. Au final, la position, l'orientation et les paramètres internes de toutes les caméras sont exprimés dans le même repère réel, ce qui garantit que, après transformation vers le repère virtuel, la caméra virtuelle est alignée avec les yeux de l'avatar reconstruit en 3D (voir figure 4.6).

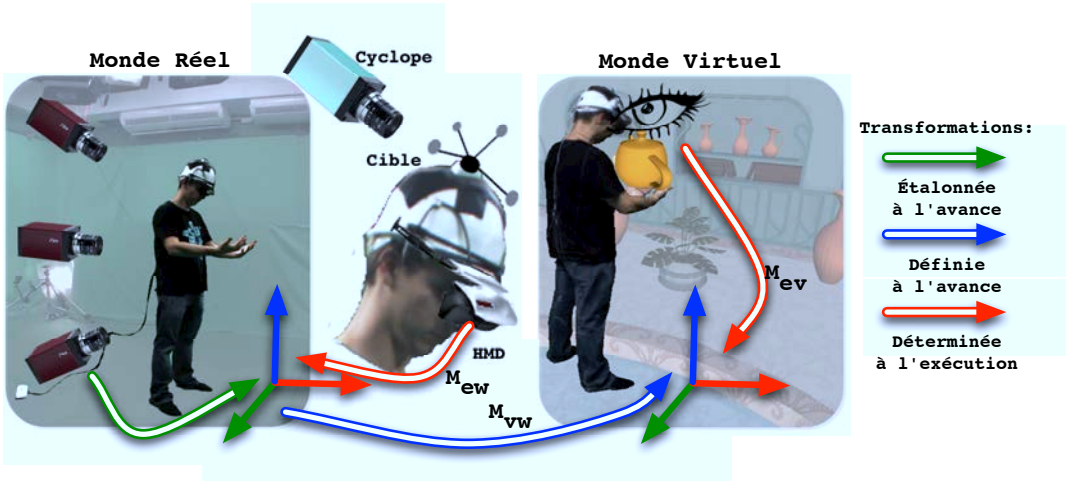


FIGURE 4.6 – Alignement des repères de la reconstruction 3D et du suivi du regard de l'utilisateur.

4.3 Mixer réel et virtuel

Une difficulté dans les applications mixant le réel et le virtuel, comme les applications de réalité augmentée, est de connaître la distance entre le corps de l'utilisateur ou le monde réel et le plan matérialisé par les écrans. En effet, pour composer l'image affichée dans le *HMD*, il faut déterminer quels objets virtuels seront cachés ou pas par le corps ou par un des objets réels présents dans le champs de vision de l'utilisateur.

Ce problème se retrouve dans les systèmes de réalité augmentée avec les *HMDs* pourvus d'écrans semi-transparents (*optical see-through*, voir figure 4.7-(a)). Ces derniers permettent à l'utilisateur de voir son propre corps au travers des écrans ainsi que des objets virtuels, ce qui est assez semblable à notre concept. D'autres systèmes utilisent une ou deux caméras montées sur le casque devant les écrans (*video see-through*, voir figure 4.7 (b)). Elles transmettent leurs images aux noeuds chargés de l'affichage dans le *HMD*, qui s'assurent de la composition entre la scène réelle observée indirectement par l'utilisateur et les objets virtuels. Dans ces deux cas on se heurte à ce problème de composition réel/virtuel [62].

Certaines applications utilisent une cible suivie par un périphérique externe, comme pour le suivi de la tête, afin de déterminer la position d'un point de la main de l'utilisateur Ceci implique les limitations citées dans la section §2.1.1.3, à savoir, le côté ponctuel de l'information et le besoin de tenir un objet ou de s'équiper les mains de gants ou de cibles autocollantes.

(a) Optical see-through (b) Video see-through

FIGURE 4.7 – Les types de *HMDs* pour la réalité augmentée.

Les systèmes qui utilisent l'information issue de la, ou des caméra(s) placée(s) devant les écrans, pour déterminer la profondeur des objets observés, se basent sur les méthodes décrites dans la section §2.1.2. Il est en effet possible :

- dans le cas où on utilise une caméra : de reconnaître une main ou un objet et d'en déduire sa distance grâce à des pré-requis sur sa taille,
- dans le cas où on utilise deux caméras : de produire une carte de profondeur et donc d'avoir une information de distance sur les objets réels.

Ces techniques permettent donc de mieux composer l'image rendue sur les écrans du *HMD* en cachant bien les parties des objets virtuels qui sont derrière les objets réels.

Elles souffrent, par contre, des limitations des systèmes mono ou stéréoscopiques (voir section §2.1.2.2), à savoir, qu'il n'est pas possible de déduire l'épaisseur des objets réels observés. Ceci limite les possibilités d'interaction et peut mener également à des incohérences graphiques : un objet virtuel peut sembler être derrière un objet réel du point de vue de l'utilisateur alors qu'il est en train de rentrer dedans. La connaissance de la géométrie complète de la scène réelle aurait permis de contrôler un périphérique à retour d'effort pour empêcher, par exemple, à la main de rentrer dans un objet virtuel, ou de piloter une simulation physique pour repousser l'objet lors d'une collision.

Par opposition, notre système est capable de mixer correctement le réel et le virtuel par le biais de la modélisation 3D. En effet, les objets réels étant *virtualisés*, il suffit de composer la scène en 3D et de l'afficher selon le point de vue de l'utilisateur (voir figure 4.8). Les problèmes de profondeur seront gérés intrinsèquement par le moteur de rendu 3D.

Il n'existe pas, à notre connaissance, de travaux sur l'intégration d'un *HMD* dans un environnement multi-caméra capable de faire de la reconstruction 3D en temps-réel pour des applications d'interaction à la première personne.

FIGURE 4.8 – Configuration de notre système avec visualisation et interaction à la première personne et mixage correct réel/virtuel.

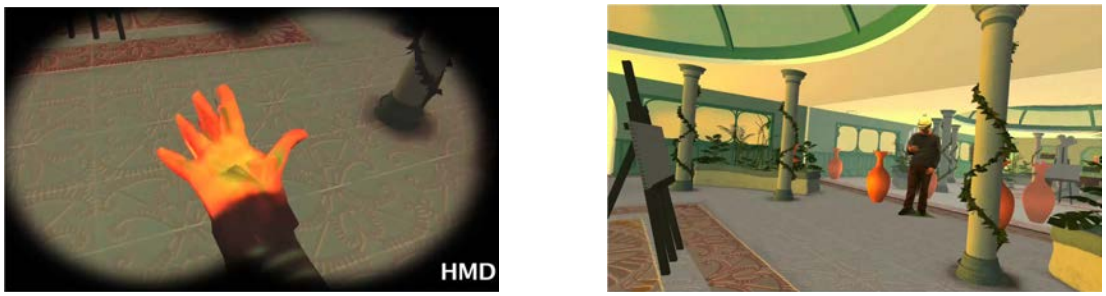
4.4 Récapitulatif

Un utilisateur muni d'un *HMD* peut donc évoluer dans notre plateforme, être reconstruit en 3D grâce au système multi-caméra et percevoir le monde virtuel au travers du casque. Grâce à un système de suivi de la tête, nous pouvons piloter le point de vue que l'utilisateur a sur la scène virtuelle. Sous réserve d'aligner correctement les repères des différents périphériques d'acquisition de données (caméras de reconstruction 3D et caméra de suivi du regard), il est possible d'aligner la vue de l'utilisateur réel avec celle de sa représentation virtuelle. On peut donc lui donner l'impression d'être « à l'intérieur » de son avatar, de ne faire plus qu'un avec lui. Il pourra, par exemple, voir ses mains reconstruites en 3D, à l'emplacement de ses propres mains et manipuler des objets virtuels comme s'il les tenait (voir figure 4.8), plus naturellement.

Nous avons donc une solution pour mixer le monde réel et l'environnement virtuel, par le biais du clone numérique, dans la limite des capacités de notre système de reconstruction 3D (pas de concavités notamment).Nous avons également un moyen pour immerger visuellement l'utilisateur dans ce monde virtuel grâce à un périphérique de visualisation unidirectionnelle qui permet des interactions et une vision à la première personne.

4.5 Implantation

L'intégration au sein de notre application a été simplifiée par l'approche modulaire de *FlowVR* (voir figure 4.10). Nous avons encapsulé le composant



(a) Vue du HMD (b) Vue externe

FIGURE 4.9 – Exemple d'immersion dans un environnement virtuel. L'utilisateur peut voir son clone numérique aligné avec son propre corps.

chargé de calculer la position de la cible dans un module *FlowVR*. Ce dernier transfère la matrice, qui transforme le repère de la cible dans le repère du *Cyclope* (M_{tc}), à un module qui gère le placement du point de vue dans la scène (M_{ev}). Ce dernier prend en argument les paramètres d'étalonnage de la caméra du *Cyclope* afin d'en déduire la position de la cible dans le même repère que celui de la modélisation 3D. Le point de vue est transmis sous forme de primitive graphique *FlowVR-Render* au module chargé de générer le rendu sur le casque.

La partie chargée de produire le modèle 3D et la partie chargée de produire un point de vue sur la scène s'exécutent de manière complètement désynchronisée. En effet, la détection de la cible peut s'effectuer à une fréquence plus importante et doit avoir la latence la plus faible possible afin de ne pas gêner l'utilisateur dans sa perception visuelle. En cas de latence, quand l'utilisateur bouge la tête, l'affichage tarde un peu à se rafraîchir en fonction du mouvement et cela peut occasionner une sensation de malaise. Nous ne pouvons donc pas synchroniser cette partie à la reconstruction 3D.

Un module sert également à régler la matrice de transformation entre le regard de l'utilisateur et celui de la cible fixée sur le casque (M_{et}). Le réglage s'effectue donc une fois que l'utilisateur a mis le casque sur sa tête, il est en effet spécifique à chaque personne. Pour faciliter le réglage, on peut placer un objet réel devant ses yeux (ses mains par exemple) et essayer d'aligner l'affichage de l'objet reconstruit avec sa vision de l'objet réel sur les bords du casque.

Une autre méthode utilisée consiste à demander à une personne externe d'utiliser une vue à la troisième personne de la scène dans laquelle est affichée un repère qui caractérise le point de vue de l'utilisateur. Il pourra essayer d'aligner ce repère avec le regard de l'avatar reconstruit en 3D en jouant sur les paramètres de la matrice.

Notre approche est simple et efficace et la précision du résultat reste bien suffisante pour nos applications. Le point de vue ainsi calculé donne vraiment

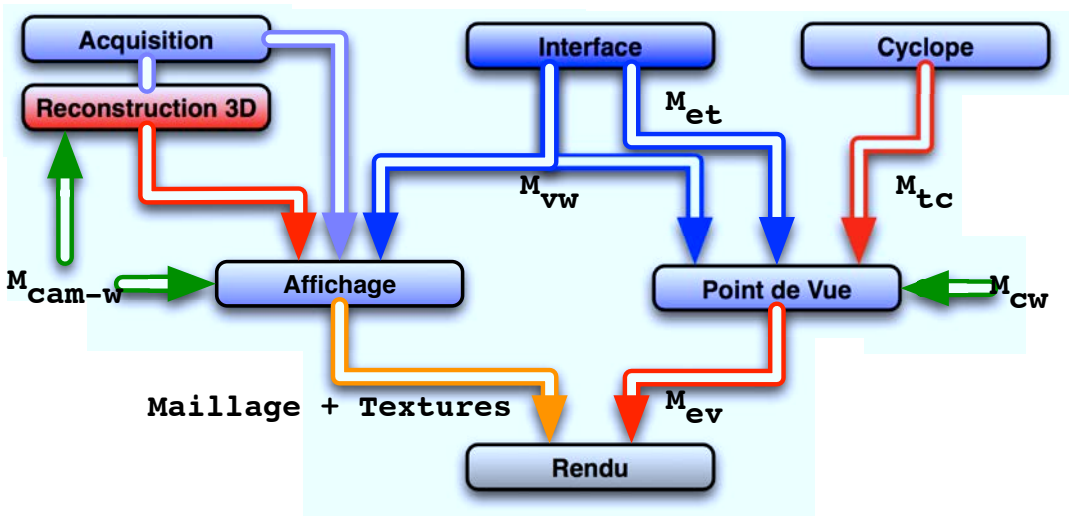


FIGURE 4.10 – Schéma de transfert des données de l'application de visualisation à la première personne.

l'illusion à l'utilisateur de voir ses mains *virtualisées* exactement à l'endroit où ses propres mains se situent. Nous pourrions mettre en place d'autres méthodes plus sophistiquées mais cela dépasse le champs d'application de nos expérimentations. Il existent des méthodes semi-automatiques où l'utilisateur doit pointer des objets virtuels avec un marqueur dont on peut connaître la position dans l'espace, et des automatiques comme celle de Gilson et al. [24], qui utilisent une étape de calibrage impliquant une caméra placée à la place des yeux de l'utilisateur et observant un damier d'étalonnage affiché sur les écrans.

Les réglages des paramètres internes de la caméra virtuelle, comme l'angle de champs de vision, ou la position du *near plane* (plan de coupe de la pyramide de vue permettant de ne pas afficher les objets trop près du centre optique), sont également réglables depuis ce module. Le champs de vue peut être calculé à partir des caractéristiques techniques du *HMD* utilisé, nous laissons tout de même la possibilité de le modifier à l'exécution. C'est un paramètre crucial pour que l'utilisateur ait une bonne perception des distances dans le monde virtuel [37].

4.6 Applications

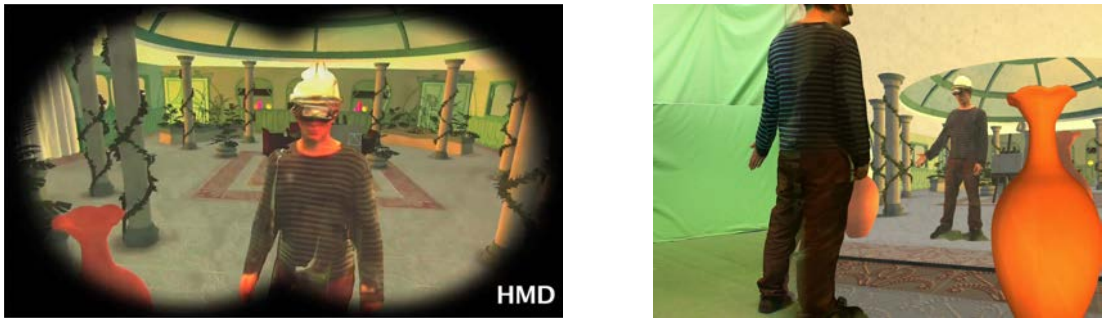
4.6.1 Miroir virtuel

Notre première idée a été de placer un miroir dans la scène virtuelle. Cela permet donc aux utilisateurs munis d'un *HMD* de se voir reconstruits en 3D dans

le miroir (voir figure 4.11). En plus de leur permettre d'apprécier la qualité de la modélisation 3D sous tous les angles, cela renforce leur sentiment de présence dans le monde virtuel.

D'après Lacan, l'image qu'un jeune enfant a de lui dans un miroir joue un rôle important dans la perception qu'il a de lui-même comme individu. De la même façon, l'image que l'utilisateur a de son avatar dans le miroir renforce son identification à cet avatar [12]. Il est d'ailleurs courant dans les logiciels de vidéo-conférence d'avoir une petite fenêtre montrant ce que voit notre propre caméra, et donc ce que voient les autres participants.

Afin de garantir la meilleure texture possible dans le reflet du miroir, nous nous arrangeons pour placer le miroir dans la scène virtuel au même endroit qu'une ou plusieurs caméras dans la scène réelle (ou inversement).



(a) Vue du HMD (b) Vue partagée réelle/virtuelle

FIGURE 4.11 – Application du miroir. L'utilisateur peut voir le reflet de son clone numérique dans le miroir virtuel.

4.6.2 Interactions

4.6.2.1 Boutons

Nous avons placé dans la scène des boutons servant à déclencher des événements. Un module gère l'activation des boutons, il reçoit en continu l'information venant de la modélisation 3D et teste si le maillage produit rentre en intersection avec un ou plusieurs boutons dont la position est définie dans un fichier de configuration. L'état de chaque bouton peut-être envoyé à d'autres modules qui peuvent utiliser cette information pour déclencher des événements.

Nous avons ainsi créé un lecteur de « vidéo 3D ». Nous avons capturé des séquences multi-caméra avec notre système et calculé une série de modèle 3D avec la version hors-ligne de notre algorithme de reconstruction 3D conjointement

(a) Vue du HMD

(b) Vue externe

FIGURE 4.12 – Application des boutons. L'utilisateur peut contrôler une video 3D en appuyant sur les boutons virtuels.

avec un atlas de texture calculé à partir des images initiales. L'utilisateur peut donc visualiser cette vidéo en 3D, et choisir son angle de vue directement en se déplaçant autour. Grâce aux boutons, il peut contrôler le lecteur, arrêter ou reprendre la vidéo, changer de séquence...

La précision du modèle 3D produit et son taux rafraîchissement de 30 à 60 Hz permettent une assez grande dextérité quant à la manipulation de ces boutons. Changer la couleur des boutons ou émettre un son lors de leur activation constitue un retour pseudo-haptique très utile à l'utilisateur.

On pourrait envisager de développer d'autres applications utilisant ces boutons, comme un instrument de musique virtuel. La version actuelle n'utilise pas l'information de vitesse du maillage mais il peut être envisagé de l'utiliser pour ajouter de l'information lors du contact, pour contrôler l'intensité d'un son, par exemple.

Un inconvénient majeur de l'utilisation de boutons virtuels dans ce type d'environnement est qu'il est aisé de les presser sans s'en apercevoir. En effet, les boutons réagissent lors de l'intersection avec n'importe quelle partie du maillage, il arrive donc qu'en explorant la scène on recule et on touche un bouton en dehors de notre champs de vision, déclenchant ainsi l'événement prévu par erreur. Des solutions à ce problème peuvent être envisagées, on peut, par exemple, activer un bouton que si l'utilisateur le regarde et que l'intersection a lieu à l'intérieur de son champs de vision, ou encore détecter les mains de l'utilisateur et définir qu'il ne peut activer un bouton qu'avec elles.

4.6.2.2 Simulation physique

En utilisant le module de simulation physique basé sur *SOFA*, présenté dans la section §2.2.4.1, nous avons enrichi le monde virtuel avec des objets déformables.

La vue à la première personne permet d'avoir une meilleure notion de la distance des objets ce qui facilite leur manipulation par rapport au système à la troisième personne présenté précédemment (voir figure 4.13).

(a) Vue du HMD (b) Vue externe

FIGURE 4.13 – Interaction avec un vase virtuel. Les collisions entre le clone numérique et le vase sont gérées par le logiciel de simulation physique SOFA.

Les interactions sont plus naturelles car elles sont co-localisées. Les objets apparaissent là où ils sont censés se situer relativement au corps de l'utilisateur. La qualité de la reconstruction 3D et sa stabilité restent des facteurs qui peuvent dégrader l'interaction basée sur la simulation physique. En effet, suivant la résolution des caméras et leur distance aux mains de l'utilisateur il est possible que les doigts ne soient pas différenciés s'ils sont trop proches ce qui peut gêner des interactions précises.

Si les interactions sont localisées à un endroit précis dans la scène virtuelle, il est possible de s'arranger pour placer les caméras de façon à avoir le bon angle de vue sur les mains de l'utilisateur et une meilleure résolution. Une autre solution sera également présentée dans le chapitre §5.

4.6.3 Démonstration

Nous avons rassemblé ces différentes applications au sein d'un démonstrateur (voir la vidéo 3) présenté lors de la conférence *Siggraph 2009* dans la partie *Emerging Technologies* [57].

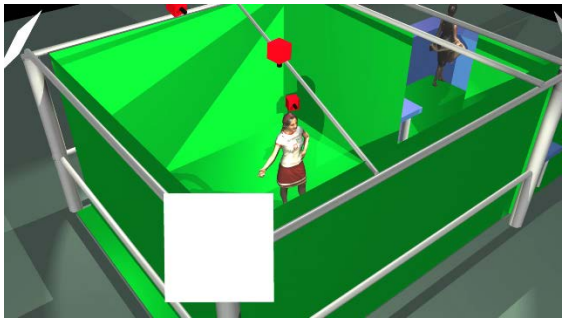
Le démonstrateur occupe un espace de la forme d'un cube de 5,5 mètres de côté recouvert de tissu vert (voir figure 4.14). Cet espace est entouré par huit caméras 2 MégaPixels et un *Cyclope* pour le suivi de la tête, ainsi que par un système d'éclairage de studio. Notre cluster se compose de quatre machines pour l'acquisition et le traitement d'images (une machine pour deux caméras), d'une machine dédiée à la reconstruction 3D, une pour la simulation physique,

et une pour le rendu, ainsi que d'un mini-PC pour le *Cyclope*. Les machines qui constituent notre cluster sont des bi-Xeon 2.66 GHz.

Le *HMD* utilisé est un prototype prêté par la société *Crescent*. Il s'agit d'un *HMD* de haute qualité contenant un écran d'une résolution de 1280x1200 pixels qui permet un angle de champs de vision horizontal de 120°. Une des particularités de ce casque vient également de son système de fixation qui assure une position relative constante entre les yeux de l'utilisateur et l'écran. Outre la rapidité de réglage et d'ajustement sur la tête, cela a l'avantage de ne pas avoir à re-régler la matrice de transformation entre les yeux de l'utilisateur et la cible à chaque utilisateur.

L'utilisateur peut évoluer dans un monde virtuel où se trouvent un miroir et interagir avec des objets déformables et avec des boutons permettant de piloter un lecteur de vidéo 3D. Il peut également voir d'autres personnes (réelles) présentes dans l'espace d'acquisition. L'espace dans lequel il est à la fois reconstruit en 3D et suivi par le *Cyclope* représente une demi-sphère d'environ 2,5 mètres de diamètre.

Plusieurs rendus sont disponibles, une copie de l'affichage du *HMD* est affichée sur un écran, ainsi qu'une vue à la troisième personne permettant aux spectateurs d'observer l'utilisateur qui évolue dans l'espace tout en voyant ce qu'il regarde. Une troisième vue est extraite directement d'une caméra afin de montrer l'utilisateur dans la salle.



(a) Modélisation 3D de notre espace de reconstruction

(b) Un participant

FIGURE 4.14 – La démonstration à la conférence *Siggraph 2009* dans la partie *Emerging Technologies*

D'une manière générale, les participants étaient très satisfaits de la démonstration et ils s'habituaient presque instantanément au monde virtuel et aux possibilités d'interactions. Le grand champs de vision et la résolution du *HMD* utilisé permettent une réelle immersion dans le monde virtuel. Seules quelques personnes ont expérimenté un inconfort à évoluer dans l'environnement virtuel, mais comme beaucoup de participants ne sont pas habitués aux applications de

réalité virtuelle immersive, ils ne peuvent pas savoir s'ils sont sujets au « mal de la réalité virtuelle ».

(a) Manipulation des boutons

(b) Utilisation du miroir

(c) 2 utilisateurs

(d) Utilisation du miroir

FIGURE 4.15 – Photos prises lors de la démonstration à *Siggraph Emerging Technologies* en 2009

Les performances du système sont bonnes, la fréquence de la reconstruction 3D oscille entre 15 et 30 Hz et sa latence est toujours inférieure à 150 ms et en moyenne de 100 ms. Cela varie en fonction du nombre de personnes présentes dans la salle et de la qualité de la soustraction de fond (qui nécessite d'être ré-initialisée toutes les heures environ). L'application s'est montrée stable et robuste sur de longues périodes.

Cette démonstration a permis d'identifier les défauts de notre système et des améliorations possibles. Un des inconvénients à utiliser un casque aussi immersif est qu'il faut adapter l'affichage en fonction du champs de vision et opter pour une projection cylindrique, ce que nous n'avions pas prévu. D'autres améliorations peuvent être envisagées, surtout dans le domaine du rendu, en effet plus la scène est réaliste, plus la sensation d'immersion est importante, une meilleure gestion des ombres aurait pu, par exemple, aider les personnes ayant du mal à se repérer dans l'espace 3D. La mise en place de la stéréovision aurait également pu contribuer à l'amélioration de l'effet de profondeur. Enfin, l'espace d'acquisition est relativement grand par rapport à nos précédentes démonstrations, ce qui nécessite de placer les caméras assez loin du centre de la pièce, ceci a eu pour conséquence de réduire la qualité du modèle et des textures produites, ce qui est particulièrement flagrant sur les mains de la personne.

4.6.4 Téléprésence

Enfin, comme présenté précédemment dans le chapitre §3, nous avons utilisé cette solution de visualisation dans notre application de téléprésence et collaboration à distance présentée dans la section §3.5.2. L'utilisateur du site de Grenoble porte donc un *HMD* et peut voir les autres utilisateurs, reconstruits en 3D ou sous la forme d'une image plaquée sur un *billboard*, comme s'ils se trouvaient devant lui. Il peut également voir le terrain sur lequel il évolue.

Cette solution de visualisation est tout à fait adaptée aux applications de téléprésence. Elle donne vraiment à l'utilisateur le sentiment d'être dans le monde virtuel, mais lui permet également d'avoir l'impression d'être physiquement avec les autres utilisateurs du monde virtuel, d'autant plus si ces derniers utilisent également un système de reconstruction 3D.

Le seul inconvénient du *HMD* est qu'il occulte une partie du visage de l'utilisateur et appauvrit un peu les émotions faciales normalement véhiculées. Il est malheureusement difficile de contourner ce problème, les *HMDs* ne couvrant pas trop la tête sont souvent moins performants en termes de résolution et de champs de vision mais peut-être que de nouvelles technologies permettront dans le futur de le résoudre. D'autres techniques peuvent être utilisées pour reconstituer la tête de l'utilisateur à l'affichage comme dans [75].

4.7 Bilan

Nous voulions améliorer notre système de reconstruction 3D et d'interaction en le couplant avec un système de visualisation unidirectionnelle à la première personne. L'utilisation d'un *HMD* pour la visualisation et du *Cyclope* pour suivre la tête de l'utilisateur s'est avéré être la meilleure solution à notre problème. L'alignement des repères des différents périphériques s'effectue lors d'une étape d'étalonnage.

Le résultat obtenu est un système où l'utilisateur peut évoluer dans un monde virtuel dans lequel sa présence est assurée par son avatar dont le modèle 3D est issu de la reconstruction 3D et les textures des images d'origine. Il peut, grâce au *HMD* et au suivi de la cible qu'il a sur la tête, visualiser le monde virtuel selon le point de vue de son avatar. Cela lui permet de regarder son corps, ses mains, par exemple, et d'interagir à la première personne de façon co-localisé.

Différentes applications ont été développées afin de lui permettre d'effectuer des interactions ou de renforcer son sentiment d'immersion dans le monde. Bien que nous n'ayons pas fait d'étude utilisateur stricte sur le système, ceci étant

en dehors de notre domaine d'expertise, il nous a semblé que les utilisateurs du système étaient plutôt satisfaits et avaient une facilité naturelle à interagir et à évoluer au sein du monde virtuel. Ceux qui avaient testé auparavant la première version du système *GrImage* ont constaté une forte amélioration du sentiment d'immersion et de la facilité d'interaction.

Des points techniques restent bien sûr à améliorer comme le rendu graphique de la scène virtuelle, l'amélioration du suivi de la tête et l'étalonnage précis du *HMD* et de la position du regard relativement à la cible. Cette plateforme ouvre également la voie à de nouvelles possibilités d'interaction qui restent à explorer. L'intégration d'un système à retour d'effort peut être envisagée même s'il n'existe actuellement aucun système permettant de couvrir tout le corps de la personne de façon dense.

D'autres améliorations pourraient être développées. Il serait intéressant, par exemple, de pouvoir de suivre la tête de l'utilisateur en utilisant le même jeu de caméras que celui utilisé pour la reconstruction 3D. Cela permettrait d'élargir l'espace dans lequel l'utilisateur peut évoluer tout en étant suivi, et pourrait améliorer la précision du suivi. De plus, il n'y aurait plus de problème d'alignement de repères. Cela implique par contre de développer des algorithmes de traitement d'images qui permettent la détection de la cible dans le domaine du visible, ou alors, de développer un système de filtre infra-rouge synchronisé avec la caméra et le flash. Avec cette dernière solution, plus technologique, une trame sur deux serait dédiée à la reconstruction 3D, l'autre au suivi de la tête.

Notre système peut également être utilisé dans un contexte multi-utilisateurs co-localisés ou distants. Un élément très limitant du *HMD* réside dans le fait que le visage de l'utilisateur est couvert à partir de son nez. Ceci est gênant pour les interactions entre utilisateurs dans un contexte de collaboration car cela occulte une partie des expressions faciales qui sont cruciales pour le dialogue.

Chapitre 5

Caméra de proximité

5.1 Motivations et problématiques

Les premiers résultats sur l'intégration d'un *HMD* avec une plateforme de reconstruction 3D multi-caméra ont été très concluants. Mais il nous est apparu que la qualité de la reconstruction 3D et des textures plaquées n'était pas toujours parfaite pour la visualisation depuis le point de vue de l'utilisateur du *HMD*.

Le modèle 3D et les données de texture sont accessibles depuis divers angles d'acquisition, ceux des caméras. Le rendu est bien indépendant du point de vue choisi, il peut être effectué depuis n'importe quelle position de caméra virtuelle. Il est donc toujours possible de rendre la scène dans le *HMD*.

Nous utilisons les images extraites des caméras pour texturer le modèle. Pour cela, nous utilisons un *shader* (programme exécuté directement sur la carte graphique) qui projette les sommets du modèle dans les images sources à l'aide des données de calibrage des caméras. Un *pixel shader* combine et lisse les contributions de chacune des caméras en prenant en compte la normale à la surface, l'angle de vue de la caméra, et la direction du regard de l'utilisateur. Le *shader* effectue également une passe pendant laquelle il détecte les occultations que l'utilisateur a avec lui-même, pour être sûr qu'une texture n'est pas appliquée sur un polygone qui est occulté par un autre polygone qui se situerait plus près de la caméra considérée. Il est donc possible que certaines facettes ne soient que peu visibles dans une ou plusieurs caméras, voire pas du tout visible sur l'ensemble des caméras, elles n'auront donc pas d'informations de texture.

Texturer le modèle 3D apporte beaucoup d'indices sur l'apparence visuelle des objets observés. Par exemple, on peut voir les plis sur la surface des vêtements alors qu'ils ne sont pas présents sur le modèle 3D d'origine. Quand on ajoute les textures, l'utilisateur portant le *HMD* perçoit un changement drastique dans son sentiment de présence dans le monde virtuel. Il peut reconnaître ses propres mains, avec par exemple, ses bijoux ou sa montre, ou encore un tatouage qu'il aurait

sur l'avant-bras. Il peut également reconnaître une autre personne entrant dans l'espace d'acquisition, ce qui est très important dans le cas de travaux collaboratifs.

Mais les caméras peuvent être assez éloignées de la surface à texturer ou non alignées avec le point de vue de l'utilisateur. Si, par exemple, il met ses mains proches de ses yeux pour avoir une vue plus détaillée, il s'aperçoit que la reconstruction 3D n'est pas très fine et que les textures ne sont pas d'une très bonne résolution. Ceci est d'autant plus vrai que la plateforme est grande, car on est forcé de placer les caméras plus loin de l'utilisateur pour augmenter leurs champs de vision sur la scène. On perd donc en résolution aussi bien pour les textures que pour la précision des silhouettes extraites, et donc du modèle 3D rendu. Les doigts d'une main peuvent ne pas être bien distincts et les textures associées donneront une impression de flou.

Un autre facteur important pour la qualité géométrique et visuelle du modèle est le placement des caméras. Avec les mains placées devant son visage, peu de caméras peuvent avoir un point de vue précis sur la partie observée, selon leur emplacement il pourra même être difficile d'en trouver une avec un angle de vue proche de celui du point de vue de l'utilisateur et pour laquelle la tête de l'utilisateur n'occulte pas la surface de la main.

5.2 Conception

Nous avons fixé une caméra sur le *HMD* (voir figure 5.1) afin de résoudre les problèmes mentionnés précédemment. Voici les étapes par lesquelles nous avons dû passer pour intégrer la caméra avec le reste du système et les bénéfices que nous avons essayé d'en tirer.

FIGURE 5.1 – Notre *HMD* expérimental avec la caméra fixée sur la visière et la cible

5.2.1 Calibrage

La caméra, fixée sur le *HMD*, est incluse dans le processus d'étalonnage du système multi-caméra, tout comme le *Cyclope*. On fixe le *HMD*, pendant cette phase, dans une position écartée du centre de l'espace d'acquisition mais où la cible est toujours visible par le *Cyclope*. Il faut également qu'il ne gène pas d'autres caméras du système, ainsi le bâton de calibrage est visible par toutes les caméras, par le *Cyclope* et par la caméra du *HMD*. Pour être sûr que les images extraites de cette caméra sont cohérentes avec celles du reste du système, elles sont toutes synchronisées ensemble (grâce à leur *gen-lock*).

Cet étalonnage permet de récupérer les paramètres intrinsèques de la caméra mais également sa position à l'initialisation. La caméra est fixée sur l'*HMD* de la même façon que la cible, utilisée pour le suivi de la tête. Sa position initiale nous permet de calculer la matrice transformant le repère de la caméra dans celui de la cible (M_{mt}) (voir figure 5.2) grâce à :

$$M_{mt} = M_{tw}^{-1}.M_{mw-init} \ . \tag{5.1}$$

Une fois que l'*HMD* devient mobile, nous calculons la position de la caméra avec :

$$M_{mw} = M_{tw}.M_{mt} \ . \tag{5.2}$$

5.2.2 Suivi du regard

Nous avons essayé de développer une autre méthode pour déterminer la position de la caméra mobile dans la scène. Cette méthode repose sur des algorithmes de vision par ordinateur qui permettent de déterminer le positionnement d'une caméra, dont les paramètres intrinsèques sont au préalable calibrés, en fonction de ce qu'elle voit.

En partant du constat que la caméra mobile observe une partie de la scène qui est également visible dans les caméras du système, et que ces dernières sont déjà calibrées, nous avons utilisé un algorithme de détection et mise en correspondance de points d'intérêts entre les images de la caméra mobile et celles du système. À partir de ces correspondances, de la position des caméras du système et de la dernière position connue de la caméra mobile, nous tentons de trouver la nouvelle position de la caméra mobile, qui minimise la distance entre toutes les paires de ligne de vue, issues des correspondances dans les images.

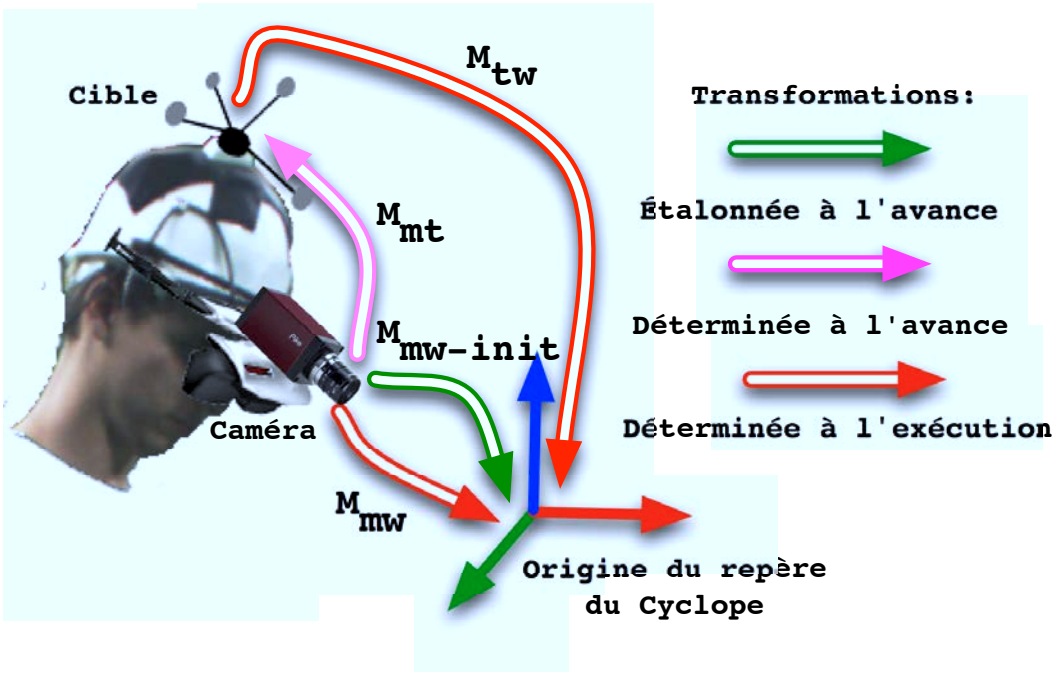


FIGURE 5.2 – Étalonnage de la caméra mobile.

Cette solution bien que prometteuse théoriquement s'est avérée difficile à mettre en pratique pour plusieurs raisons :

- Notre environnement est relativement uni, il est donc compliqué de détecter des points d'intérêts dans les images. Or, il faut un minimum de 6 correspondances pour trouver une solution au système non linéaire.
- En cas de mouvement rapide de l'utilisateur, les images issues de la caméra mobile sont floues ce qui rend la détection de points d'intérêts moins évidente.
- La résolution du système non linéaire par optimisation ne tourne pas en temps-réel.

Des travaux sont en cours pour proposer une solution plus efficace et plus précise.

5.2.3 Texture

Grâce à la position calculée en temps-réel et à la matrice des paramètres intrinsèques du calibrage, nous pouvons en déduire la matrice de projection de la caméra pour chaque itération du *Cyclope*. Il nous suffit donc de projeter le modèle 3D dans l'image de la caméra, en utilisant la matrice de projection de la caméra que nous venons de calculer, pour récupérer les bonnes coordonnées de texture des sommets.

Ceci permet de texturer le modèle 3D :

- soit uniquement avec la texture de la caméra de proximité,

- soit avec un mélange entre la texture de la caméra de proximité et les textures issues des caméras les plus proches des facettes considérées et ayant un angle de vue proche de la normale aux facettes et de l'angle de vue de l'utilisateur.

Cette approche permet d'éviter d'avoir à effectuer une étape de soustraction de fond pour la caméra mobile. Comme la caméra bouge, il est difficile de se baser sur une méthode d'apprentissage du fond.

5.2.4 Reconstruction 3D

Afin d'obtenir un modèle 3D plus précis des objets observés de près, comme les mains de l'utilisateur par exemple, il est possible de se servir de la caméra mobile au sein de l'algorithme de reconstruction 3D. Cela impose plusieurs contraintes.

Il faut être capable d'effectuer une soustraction de fond sur l'image de la caméra mobile afin d'obtenir une silhouette de la main. Ceci peut-être effectué avec une technique de *chromakey* : Nous connaissons a priori la couleur du fond de notre plateforme et pouvons ainsi segmenter les objets présents dans l'image rapidement et éliminer les pixels à peu prêt de cette couleur. La limitation de cette technique est qu'il faut un fond absolument de la même couleur et un éclairage très maitrisé. Ceci est rarement le cas, les caméras ou leurs pieds seront, par exemple, considérés comme ne faisant pas partie du fond. Mais, si cette technique n'est utilisée que pour une caméra, la reconstruction 3D ne devrait pas en souffrir car ces parties non désirées ne seront présentes que dans une seule silhouette et seront donc absentes du modèle 3D. Ainsi cela permettrait de différencier les doigts de la main de l'utilisateur.

Une autre contrainte amenée par l'utilisation de la caméra mobile pour la reconstruction 3D vient du fait que cette caméra va réduire l'espace de reconstruction 3D. En effet, l'espace de reconstruction 3D est l'intersection des pyramides de vue de toutes les caméras. La caméra de proximité devient la caméra qui limite le plus l'espace de reconstruction et il ne peut donc pas y avoir de modèle 3D reconstruit en dehors de son champs de vue. Ceci ne pose pas de problème quand les interactions ne se situent qu'au niveau de ce que l'utilisateur regarde et que aucune visualisation de l'utilisateur en entier n'est nécessaire. Dans le cas contraire il faut trouver une solution.

Une possibilité est de faire deux reconstructions 3D, l'une incluant toutes les caméras (dont la caméra mobile) pour les interactions précises situées dans le champs de vision de la caméra mobile et pour la visualisation dans le *HMD* et l'autre sans la caméra mobile pour une visualisation externe (ou dans un miroir) et pour les interactions en dehors du champs de vision. Ceci est assez lourd mais

tout à fait envisageable. De plus, le champs de vision de la caméra mobile est plus restreint que celui du *HMD* ce qui peut limiter la vue de l'utilisateur.

Une autre possibilité est d'étendre virtuellement le champs de vue de la caméra mobile. Pour cela il suffit de créer une silhouette virtuelle dont la résolution est supérieure à celle de l'image de la caméra mobile. Cette silhouette contiendra la silhouette d'origine en son centre et sera complétée par un masque indiquant à la reconstruction 3D de considérer la partie en dehors de l'image initiale comme faisant partie de la silhouette (voir figure 5.3). Il pourrait également être possible de ré-implémenter l'algorithme *EPVH* pour traiter ce cas à part.

FIGURE 5.3 – Création d'une silhouette virtuelle pour étendre virtuellement le champs de vision de la caméra mobile.

Enfin, il est impératif d'avoir une gestion de la synchronisation très précise. Il faut synchroniser non seulement les caméras du système avec la caméra mobile, ce qui est facilement réalisable au niveau du matériel, mais également synchroniser le *Cyclope* avec tout le système. En effet, il est nécessaire que toutes les silhouettes réfèrent au même instant, mais également que les informations concernant la projection de la caméra mobile soient datées de cet instant. Il en va de la précision du modèle 3D produit. Cette synchronisation sera développée dans la section §5.3.2.

5.3 Expérimentations

5.3.1 Texture

5.3.1.1 Implantation

L'implantation du module qui gère le placage de la texture issue de la caméra mobile nécessite juste d'adapter le module en charge du placage de texture. Ce dernier doit, en effet, recevoir les textures issues de la caméra mobile ainsi que la matrice de projection calculée à partir de la matrice de transformation venant du *Cyclope* (M_{cw}) et de la matrice intrinsèque de la caméra calculée à l'initialisation. L'intégration avec le reste du réseau se fait comme indiqué sur la figure 5.4. La barrière de synchronisation est nécessaire pour assurer une cohérence temporelle entre les données issues des caméras fixes, celles issues de la caméra mobile et celles issues du traqueur 3D.

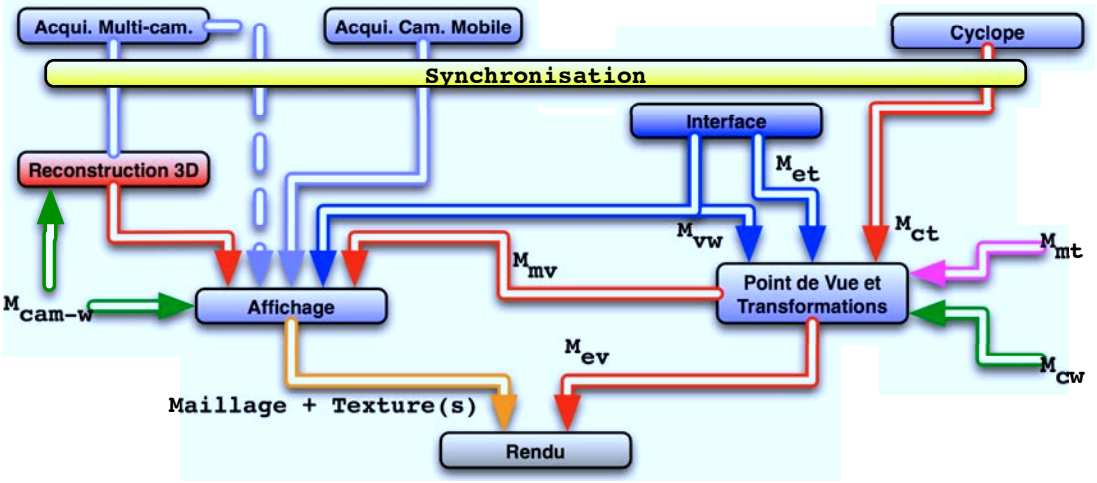


FIGURE 5.4 – Schéma de transfert des données de l'application de placage de texture issue de la caméra mobile.

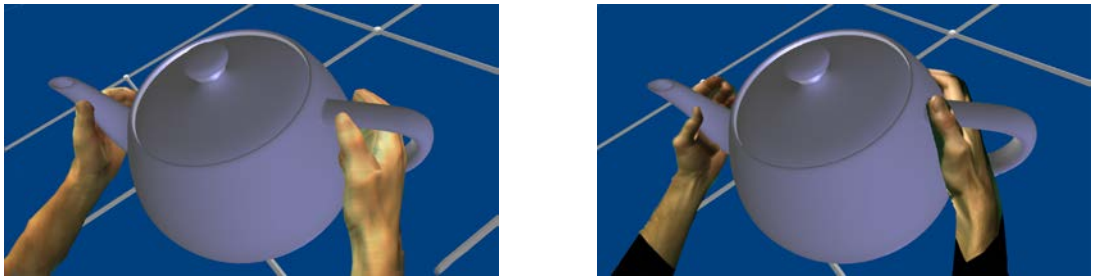
5.3.1.2 Résultats

Les résultats montrent que chacune des méthodes a des avantages et des inconvénients en matière de qualité visuelle.

1. La première méthode utilisée avant l'implantation de la caméra mobile, utilise les textures issues des images des caméras fixes. Les textures présentes sont floues et de basse résolution, mais le résultat est quand même acceptable car

bien plaqué sur le modèle 3D reconstruit (voir figure 5.5-(a)). Ces problèmes sont dus au fait que les caméras ne sont pas forcément bien alignées avec le point de vue de l'utilisateur et que les textures sont mélangées.

2. La nouvelle méthode, dans la mesure où elle n'utilise que la caméra mobile pour texturer le modèle 3D, présente une texture plus nette et de meilleure résolution, obtenue grâce à la proximité de la caméra (voir figure 5.5-(b)). Mais du fait que le modèle 3D est une enveloppe visuelle, il inclut l'objet réel. Sa re-projection dans l'image d'origine peut donc être plus grande que la forme réelle de l'objet dans l'image. Dans ce cas, l'espace entre la silhouette réelle et la re-projection est rempli avec des pixels qui figurent le fond. Une solution robuste et durable dans le temps pour contrer ce défaut serait de développer un algorithme de soustraction de fond temps-réel dans un contexte mobile.



(a) Textures venant de toutes les caméras (b) Texture venant de la caméra mobile

FIGURE 5.5 – Test de placage de texture dans différentes configurations.

Nous n'avons pas comparé le sentiment des utilisateurs face à ces deux approches mais généralement ils préfèrent des textures de qualité alignées avec leur vue, même si elles comportent des pixels issus du fond, plutôt qu'un mélange de plusieurs textures. De plus, la visualisation est plus stable quand l'objet bouge. Sans la caméra mobile, les textures sont mélangées ensemble, si les objets bougent la texture dominante peut changer et le résultat donnera l'impression de scintiller un peu. De plus, la texture issue de la caméra mobile peut masquer des changements rapides de géométrie. Des petits mouvements peuvent résulter en des changements abruptes d'orientation de facettes, souvent dues à des imprécisions numériques. Dans le cas où c'est la tête de l'utilisateur qui bouge rapidement, les images issues de la caméra mobile ont tendance à être floues.

Nos premiers essais, dans un contexte applicatif, nous ont montrés que cette méthode est très sensible à l'étalonnage initial de la caméra mobile et sa robustesse au cours du temps. La caméra et la cible doivent être fixées ensemble de façon très rigide et l'étalonnage hors-ligne et en-ligne doivent être très précises afin de ne pas mener à des erreurs de re-projection et à des incohérences dans les textures.

En effet, cette méthode de calibrage en deux temps accumule plusieurs sources d'erreur. La caméra que nous utilisons pour le suivi de la cible ne s'est pas montrée

assez précise. Cette imprécision étant présente lors de l'étalonnage initial et lors du suivi temps-réel de la cible, la matrice de transformation de la caméra mobile et donc la matrice de projection utilisée pour projeter la texture sur le modèle accumule beaucoup trop d'erreur. L'utilisation d'un système commercial plus fiable, basé sur plus d'une caméra pourrait éventuellement réduire cette erreur et rendre la caméra mobile plus fonctionnelle dans un cadre applicatif.

5.3.2 Reconstruction 3D

5.3.2.1 Implantation

L'intégration de la caméra mobile dans le processus de reconstruction 3D est assez simple. Il faut pour cela ajouter l'image issue de la caméra mobile au pipeline de traitement d'images, à la différence que la soustraction de fond sera uniquement basée sur la couleur des pixels et non sur un apprentissage de modèle du fond. De plus, il faut pouvoir changer dynamiquement la matrice de projection de la caméra mobile dans le processus de reconstruction 3D (voir figure 5.6).

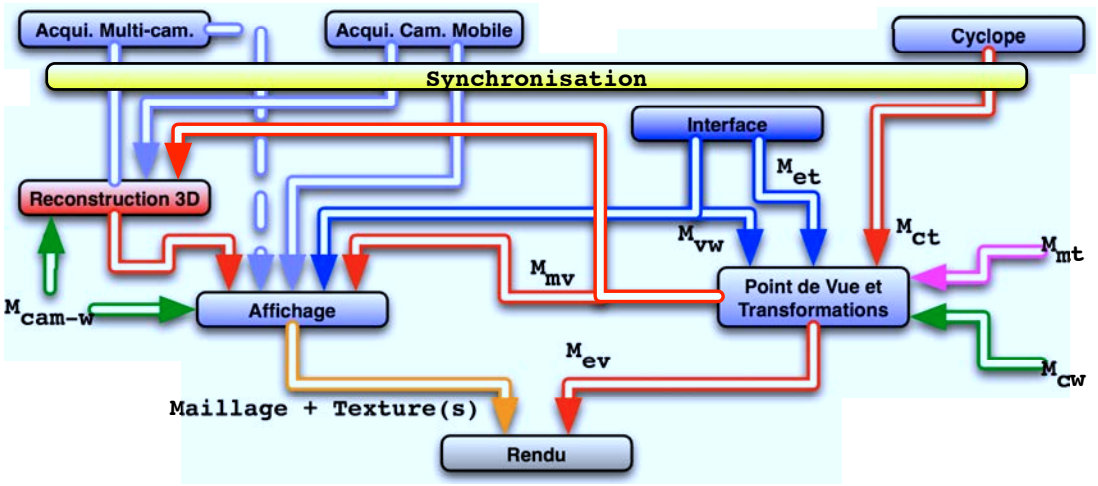


FIGURE 5.6 – Schéma de transfert des données de l'application de reconstruction 3D avec la caméra mobile.

5.3.2.2 Résultats

Nous avons pu démontrer la faisabilité d'une reconstruction 3D à l'aide d'une caméra de proximité pour obtenir plus de détail sur une partie du modèle 3D mais les résultats obtenus dans un contexte mobile ne sont pas satisfaisants. Ceci pour deux raisons :

1. La soustraction de fond dans un contexte mobile n'est pas efficace et les silhouettes obtenues n'apportent pas d'information d'assez bonne qualité pour améliorer le modèle. Un environnement d'un vert uniforme ou un algorithme capable de soustraire le fond dans un contexte mobile avec fond non uniforme sont nécessaires au bon fonctionnement de cette méthode.
2. Comme pour le placage de texture, l'accumulation d'erreur sur la matrice de projection de la caméra tend à donner des modèles incohérents et de moins bonne qualité que les modèles obtenus sans prendre en compte la caméra mobile dans le processus de reconstruction 3D. Un système de suivi de cible plus précis et robuste dans le temps pourrait améliorer ce facteur de qualité.

5.4 Bilan

Nous avons donc montré que l'ajout d'une caméra à des fins d'amélioration visuelle et mécanique du modèle 3D observé par l'utilisateur est intéressant. Il reste pourtant des limitations techniques à son utilisation dans un contexte applicatif, à savoir la nécessité d'avoir un système de suivi de cible précis et robuste dans le temps et l'espace, et de développer une méthode de soustraction de fond qui ne soit pas basée sur l'apprentissage d'un fond fixe.

Si ces limitations techniques sont levées, il sera alors possible de modéliser avec précision les objets que l'on regarde (ou ses propres mains) et d'en avoir une visualisation de bonne qualité. Il est également envisageable d'utiliser deux caméras afin d'affiner le modèle 3D avec des techniques de stéréovision.

Une étude utilisateur est à prévoir pour évaluer correctement l'apport de la visualisation à la première personne et de l'ajout de la caméra mobile sur le sentiment de présence de l'utilisateur et sa dextérité.

III

Champs de vitesse

« Never confuse movement with action. »
Ernest Hemingway

Chapitre 6

Les champs de vitesse

6.1 Motivations et problématique

L'algorithme EPVH calcule un nouveau modèle 3D à chaque itération. Il ne fournit pas d'informations de mouvement sur la géométrie acquise. Ceci peut être une limite pour certaines applications. Par exemple, une force de réaction ne peut être modulée en fonction de la vitesse du modèle 3D lors de la détection d'une collision avec un objet virtuel. Il est également compliqué d'attacher un objet virtuel à une partie de la géométrie. De plus, une information de mouvement dense est une étape intermédiaire fondamentale de la chaîne de traitement d'images, sur laquelle peuvent être développées des applications de plus haut niveau comme, par exemple, du suivi de maillage ou de la segmentation, qui elles mêmes peuvent servir pour du transfert ou de la reconnaissance d'actions.

Pour cela, l'observation des pixels, issus des images, fournit des informations utiles sur le mouvement, à travers les variations temporelles de la fonction d'intensité. Dans une configuration mono-caméra, ces variations permettent d'estimer des champs de vitesse 2D denses dans l'image : le *flot optique*. L'estimation du flot optique a été un sujet d'intérêt dans la communauté de la vision par ordinateur ces dernières dizaines d'années et de multiples méthodes ont été proposées [6, 30, 46]

Dans le cas d'un système de caméras, l'intégration depuis les différents points de vue permet de considérer le mouvement des points 3D de la surface observée et d'estimer le champs de vecteur de déplacement 3D : le *flot de scène* [77, 54]. Pourtant, autant en 2D qu'en 3D, l'information de mouvement ne peut pas être déterminée indépendamment pour chaque point avec pour seule information la variation de la fonction d'intensité : une contrainte additionnelle doit-être introduite, par exemple, une hypothèse de continuité du champs de mouvement. De plus, du fait de l'approximation des dérivées par la méthode des différences finies, l'estimation du flot est connue pour être limitée à de petits déplacements. Bien que plusieurs approches en 2D aient été proposées pour faire face à ces limitations [81], moins d'efforts ont été consacrés au cas de la 3D.

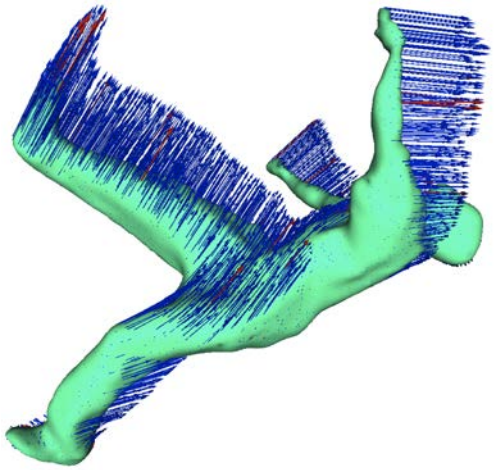


FIGURE 6.1 – Exemple de flot de scène dense (en bleu) calculé à partir de correspondances de points d'intérêts 2D et 3D (en rouge) et de flot de normal dense.

Lors de ma thèse, nous avons étudié la façon d'intégrer, de manière efficace, diverses contraintes pour estimer des informations de mouvements denses instantanés sur des surfaces 3D, à partir des variations temporelles de la fonction d'intensité issue de plusieurs images. Notre motivation première a été de fournir des indices de mouvement robuste qui peuvent être directement utilisés par une application interactive, ou qui peuvent être introduits dans des applications plus avancées comme le suivi de surface ou la segmentation. Bien que notre but ait été d'intégrer le calcul des champs de vitesse avec notre application de reconstruction 3D, l'approche n'est pas limitée à un scénario spécifique et fonctionne pour à toute application qui peut bénéficier d'une information de mouvement de bas niveau.

La plupart des approches existantes qui estiment le flot de scène font l'hypothèse des petits déplacements entre les instants de temps pour lesquels les approximations aux différences finies des dérivés temporelles sont valides. Cependant, cette hypothèse est souvent incorrecte avec les systèmes d'acquisition actuels et des objets réels en mouvement. En effet, l'amplitude des mouvements observés et la fréquence d'acquisition utilisée ne permettent pas d'effectuer cette hypothèse dans tous les cas.

Pendant ma thèse, nous avons construit une méthode unifiée permettant de lier de manière cohérente les contraintes visuelles, issues des images consécutives temporellement, avec des contraintes de déformation de surface. Pour traiter les grands déplacements, nous utilisons des mises en correspondances temporelles entre les images issues d'une même caméra. Ces contraintes agissent comme des points d'ancrage pour les régions de la surface où les déplacements sont plus importants et où les informations de variation d'intensité ne sont pas utiles. Ces contraintes visuelles sont diffusées sur la surface grâce à un schéma *Laplacien* qui régularise les vecteurs de déplacements estimés entre les points voisins de la

surface. Un élément clef de cette méthode est qu'elle conduit a des optimisations linéaires ce qui permettrait, à terme, une implémentation temps-réel.

6.2 État de l'art

Dans un article fondateur sur le flot de scène, Vedula et al. [77] explicitent la contrainte de flot normal qui lie les dérivés de la fonction d'intensité dans les images au flot de scène des points 3D de la surface. Comme mentionné précédemment, ces contraintes ne permettent pas d'estimer le flot de scène de façon indépendante à un point de la surface, des contraintes supplémentaires doivent être introduites. Au lieu d'utiliser la contrainte de flot normal, un algorithme est proposé qui estime de façon linéaire le flot de scène à partir de la géométrie 3D de la surface et du flot optique 2D. Le flot optique permet de mieux contraindre le flot de scène que le flot normal, mais son estimation est fondée sur des hypothèses de lissage qui tiennent rarement dans le plan image mais sont souvent vérifiées dans le cas de surfaces.

Dans [54], Neumann et Aloimonos introduisent un modèle de subdivision de surface qui permet d'intégrer sur la surface, les contraintes de flot normal avec des contraintes de régularisation. Néanmoins, cette solution globale suppose encore de n'être en présence que de petits mouvements et peut difficilement faire face à des cas comme ceux présentés dans nos expérimentations.

Une autre stratégie est suivie par Pons et al. [61], qui ont présenté une approche variationnelle qui optimise un critère de consistance photométrique au lieu des contraintes de flot normal. L'intérêt est que la cohérence spatiale comme la cohérence temporelle peuvent être appliquées, mais au prix d'une optimisation coûteuse en calcul. Au contraire, notre objectif n'est pas d'optimiser la forme observée, mais de fournir une information de mouvement dense de façon efficace et rapide.

Plusieurs travaux [85, 32, 79] considèrent le cas où la structure de la scène est décrite par une carte de disparité issue d'un système stéréoscopique. Ils proposent l'estimation combinée de la disparité spatiale et temporelle du mouvement 3D. Nous considérons une situation différente dans laquelle la surface de la forme observée est connue, par exemple, un maillage obtenu en utilisant une approche multi-vues. Ceci permet une régularisation du champ de déplacement sur un domaine où les hypothèses de régularité sont vérifiées.

Il convient de mentionner également les approches récentes sur le suivi temporel de surface [72, 76, 53, 8] qui peuvent également fournir des champs de vitesse. C'est en effet une conséquence de la mise en correspondance de surfaces dans le

temps. Notre but est ici différent, notre méthode ne fait aucune hypothèse sur la forme observée, seulement quelques hypothèses sur le modèle de déformation locale de la surface. Notre méthode fournit des informations bas niveau, le mouvement instantané, qui peuvent à leur tour être utilisées comme données d'entrée d'une méthode d'appariement ou de suivi de surface.

Nos contributions à l'égard des approches mentionnées sont de deux ordres :

- En suivant les travaux sur l'estimation robuste du flot optique 2D [44, 81], nous utilisons avantageusement les valeurs de déplacement robuste fournies par le suivi de points d'intérêts dans des images consécutives dans le temps. Ces points intérêts permettent de contraindre les grands déplacements alors que les contraintes de flot de normal permettent de modéliser précisément les déplacements les plus petits.
- Une résolution linéaire combine ces différentes contraintes visuelles avec un modèle de déformation de surface et permet une résolution itérative ainsi qu'un raffinement de type multi-échelle.

6.3 Définitions préliminaires

Notre méthode peut s'appliquer en sortie de tout système multi-caméra capable de produire un flux de surface se déplaçant de manière non rigide. Chacune de ces surfaces ayant été modélisée indépendamment de la précédente à partir des N caméras calibrées du système. Toutes les méthodes de reconstruction 3D décrites dans la section §2.1.2.3 peuvent être utilisées.

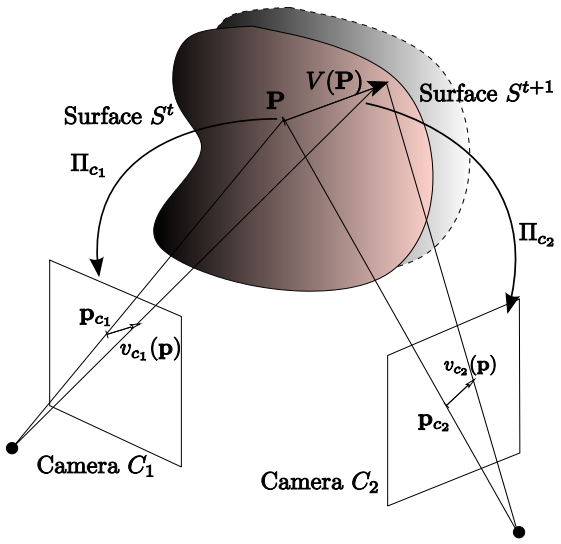


FIGURE 6.2 – Projection du flot de scène en flot optique sur les différentes images issues du système multi-caméra.

La surface au temps t est dénotée $\mathcal{S}^t \subset \mathbb{R}^3$ et associée au jeu d'images $\mathcal{I}^t = \{\mathbf{I}_c^t \mid c \in [1..N]\}$. Un point 3D $\mathbf{P}$ sur la surface est décrit par le vecteur $(x, y, z)^T \in \mathbb{R}^3$. Sa projection dans l'image $\mathbf{I}_c^t$ est le point 2D $\mathbf{p}_c$ qui a comme coordonnées $(u_c, v_c)^T \in \mathbb{R}^2$, calculées en utilisant la matrice de projection 3x4 $\mathbf{\Pi}_c$: $\mathbb{R}^3 \mapsto \mathbb{R}^2$ de la caméra c (voir figure 6.2). La région 3D de l'image correspondant à la visibilité de $\mathcal{S}^t$ dans $\mathbf{I}_c^t$ est notée $\Omega_c^t = \mathbf{\Pi}_c \mathcal{S}^t$.

Notre méthode recherche le meilleur champ de déplacement 3D de la surface entre le temps t et $t+1$, noté $V^t : \mathcal{S}^t \mapsto \mathbb{R}^3$ avec $V^t(\mathbf{P}) = \frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t}\ \forall\ \mathbf{P} \in \mathcal{S}^t$. Ce champ de déplacement est contraint par :

- les données d'entrée comme le jeu d'images calibrées $\mathcal{I}^t$ et $\mathcal{I}^{t+1}$, et la surface $\mathcal{S}^{t+1}$,
- un modèle de déformation.

La projection du champ de déplacement 3D sur $\mathbf{I}_c^t$ est décrit par v_c^t. La relation entre un petit déplacement à la surface de $\mathcal{S}^t$ et son image prise par la caméra c est décrite par la matrice *Jacobienne* 2x3 $J_{\mathbf{\Pi}_c}(\mathbf{p}_c) = \frac{\partial \mathbf{p}_c}{\partial \mathbf{P}}$, tel que $v_c^t = J_{\mathbf{\Pi}_c}(\mathbf{p}_c) V^t$.

Dans les sections suivantes, j'expliquerai les contraintes visuelles et géométriques décrites par les données d'entrée et le modèle de déformation utilisé pour propager le mouvement sur la surface.

6.4 Contraintes visuelles

Notre méthode peut utiliser trois types de contraintes visuelles pour estimer le déplacement 3D :

1. des correspondances éparses de points d'intérêts 3D,
2. des correspondances éparses de points d'intérêts 2D et,
3. des contraintes denses de flot normal dans les images.

Chacune de ces contraintes mènera à un terme dans notre fonctionnelle d'erreur qui sera présentée dans la section §6.5.2 et qui décrit comment le champ de déplacement estimé se rapporte aux observations. Ces contraintes n'incluent pas de cohérence photométrique spatiale ou temporelle car ces dernières impliquent des termes non linéaires dans la fonctionnelle d'erreur. Elles sont plus adaptées aux problèmes liés à l'optimisation de la forme de la surface qu'à l'estimation plus bas niveau du mouvement.

6.4.1 Correspondances 3D éparses

La mise en correspondance de points d'intérêts 3D permet de recueillir de l'information pour un jeu de points 3D à la surface de $\mathcal{S}^t$. Ces points 3D et leurs déplacements associés sont obtenus par la détection des points d'intérêts 3D sur $\mathcal{S}^t$ et $\mathcal{S}^{t+1}$, en leur créant un descripteur et en les appariant grâce à la comparaison de ces descripteurs.

Il existe différentes voies pour obtenir des correspondances 3D entre deux formes. Dans notre approche, nous utilisons MeshDOG pour détecter des points d'intérêts 3D et MeshHOG pour les décrire [83]. Cette méthode définit et met en correspondance les extremas locaux de n'importe quelle fonction scalaire définie sur la surface. D'autres techniques de détection et mise en correspondance peuvent être utilisées (tel que [71]), tant qu'elles récupèrent un ensemble de correspondances robustes entre les deux surfaces.

L'avantage de l'utilisation de points d'intérêts 3D est que, contrairement au flot optique (décrit à la section §6.4.3), ils permettent de contraindre le mouvement même lors de grands déplacements dans l'espace 3D.

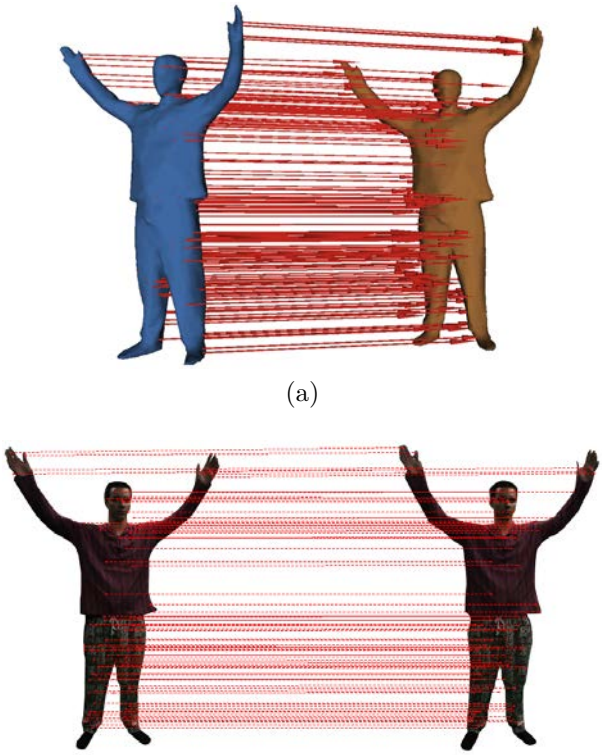
On obtient un ensemble éparse de déplacements 3D V_m^t pour des points 3D $\mathbf{P}_m \in \mathcal{S}^t$ (voir figure 6.3-(a)). Ces point forment un sous-ensemble discret de $\mathcal{S}^t$ appelé $\mathcal{S}_m^t$. La fonction d'erreur suivante décrit la proximité du champ de déplacement calculé V^t au champ de déplacement éparse V_m^t :

$$\mathbf{E}_{3D} = \sum_{\mathcal{S}_m^t} \|V^t - V_m^t\|^2 \ . \tag{6.1}$$

6.4.2 Correspondances 2D éparses

Dans notre approche, nous considérons des correspondances 2D éparses entre les jeux d'images $\mathcal{I}^t$ et $\mathcal{I}^{t+1}$. Comme dans le cas de la 3D, il y a différentes techniques existantes pour calculer des correspondances 2D entre une paire d'images, par exemple, SIFT, SURF ou Harris. Sans pour autant perdre en généralité, nous nous appuyons sur le détecteur et descripteur SIFT [45]. Il s'est avéré robuste et bien adapté dans notre cas, car invariant aux rotations et aux changements d'échelles.

Nous calculons des points d'intérêts sur les flux d'images $\mathcal{I}^t$ et $\mathcal{I}^{t+1}$. Nous mettons ensuite en correspondance les points d'intérêts venant des images $\mathbf{I}_c^t$ avec ceux de $\mathbf{I}_c^{t+1}$. Cela nous donne un jeu de déplacements 2D épars $v_{c,s}^t$ pour quelques



(a)

(b)

FIGURE 6.3 – Correspondances 3D éparses entre deux surfaces (a) et correspondances 2D éparses entre deux images (b).

points 2D $\mathbf{p}_{c,s} \in \Omega_c^t$ (voir figure 6.3-(b)). Ces point forment un sous-ensemble de Ω_c^t appelé $\Omega_{c,s}^t$. La fonction d'erreur suivante décrit la proximité du champ de déplacement 2D calculé v_c^t au champ de déplacement 2D épars $v_{c,s}^t$:

$$\mathbf{E}_{2D} = \sum_{c=1}^{N} \sum_{\Omega_{c,s}^t} \|v_c^t - v_{c,s}^t\|^2 ,$$

ce qui est équivalent à :

$$\mathbf{E}_{2D} = \sum_{c=1}^{N} \sum_{\Omega_{c,s}^t} \|J_{\mathbf{\Pi}_c} V^t - v_{c,s}^t\|^2 . \tag{6.2}$$

Il est important de noter que des correspondances 3D peuvent être obtenues à partir des points d'intérêts 2D en re-projetant les points détectés depuis $\mathcal{I}^t$ sur la surface $\mathcal{S}^t$ et ceux de $\mathcal{I}^{t+1}$ sur la surface $\mathcal{S}^{t+1}$. Cela fournit une liste de points d'intérêts 3D qui peuvent être mis en correspondance grâce à leurs descripteurs SIFT. Au lieu de réaliser cette mise en correspondance seulement dans l'espace

d'une seule image, cela permet de prendre en compte les descripteurs issus de plusieurs images. Dans ce cas, la fonctionnelle d'erreur est similaire à $\mathbf{E}_{3D}$ décrite dans l'équation (6.1).

Bien que les points d'intérêts 3D soient plus robustes, en particulier aux occultations, et fournissent une meilleure information sur de longues séquences, ils présentent des désavantages, par rapport aux points d'intérêts 2D, pour l'estimation du flot de scène. Ils ne sont pas robustes aux changements de topologie et sont plus demandeurs en puissance de calcul, ce qui peut être crucial dans certaines applications. De plus, la surface S^{t+1} est forcément requise ce qui peut être problématique pour des applications interactives.

6.4.3 Flot normal dense

Contrairement aux contraintes éparses données par les correspondances 2D et 3D, des informations denses sur V^t peuvent être obtenues en utilisant le flot optique accessible dans les images. En effet, en prenant comme hypothèse que l'illumination reste constante entre $\mathbf{p}_c^{t+1}$ et $\mathbf{p}_c^t$, la projection du même point de la surface entre deux trames consécutives, on peut définir l'équation du *flot normal* [6] comme étant :

$$\nabla I_c^t . v_c^t + \frac{\mathrm{d}I_c^t}{\mathrm{d}t} = 0\,,$$

ou équivalent en 3D à [77] :

$$\nabla I_c^t . \left[J_{\mathbf{\Pi}_c} V^t \right] + \frac{\mathrm{d}I_c^t}{\mathrm{d}t} = 0\,.$$

La fonction d'erreur suivante décrit comment le champs de déplacement 2D calculé v_c^t vérifie la contrainte de flot normal :

$$\mathbf{E}_{flow} = \sum_{c=1}^{N} \int_{\Omega_c^t} \| \nabla I_c^t . \left[J_{\mathbf{\Pi}_c} V^t \right] + \frac{\mathrm{d}I_c^t}{\mathrm{d}t} \|^2 \, \mathrm{d}\mathbf{p}_c\,. \tag{6.3}$$

Cependant, cela ne contraint les déplacements 2D que dans la direction du gradient de l'image ∇I_c^t, qui est la composante normale du flux optique. Comme il sera expliqué dans la section suivante, c'est le *problème de l'ouverture* connu en 2D et étendu à la 3D.

6.5 Régularisation

Les correspondances éparses 2D et 3D contraignent seulement le déplacement de la surface pour des points 3D spécifiques et pour leur re-projection dans les images. Pour trouver un champ de mouvement dense sur la surface, nous avons besoin de propager ces contraintes en utilisant un terme de régularisation.

En outre, comme mentionné précédemment, les contraintes denses de flot normal ne fournissent pas assez de contraintes pour estimer les déplacements 3D. En effet, il peut être démontré que les équations du flot normal pour des projections dans différentes images d'un même point 3D $\mathbf{P}$ contraignent de façon indépendantes V^t à $\mathbf{P}$, et ne résolvent donc que 2 degrés de liberté sur 3. Vedula et al. [77] mentionnent deux stratégies de régularisation pour faire face à cette limitation. La régularisation peut être effectuée dans les plans images en estimant les flux optiques qui fournissent des contraintes plus complètes sur le flot de scène, ou elle peut être effectuée sur la surface 3D.

Puisque nous avons connaissance de la surface 3D et que les contraintes éparses 2D et 3D doivent être également intégrées, un choix naturel dans notre contexte est de régulariser en 3D. En plus, la régularisation dans l'espace image souffre d'artefacts et d'incohérences résultant des discontinuités de profondeur et des occultations qui contredisent l'hypothèse de lissage, alors qu'une telle hypothèse se justifie sur la surface 3D.

6.5.1 Modèle de déformation

Les hypothèses de régularité sur les champs de déplacement 3D de la surface limitent les déformations de cette surface à un niveau local. Elles définissent ainsi un modèle de déformation de la surface, par exemple, une rigidité locale. En 2D, de nombreuses méthodes de régularisation ont été proposées pour l'estimation du flot optique, elles se répartissent en 2 grandes catégories : les régularisations locales ou globales. Elles peuvent être étendues à la 3D.

Par exemple, la méthode 2D de Lucas et Kanade, qui utilise un voisinage local, a été appliquée en 3D par Devernay et al. [18]. Toutefois, le modèle de déformation associé à la surface n'a pas de signification réelle, car les contraintes de déformation ne se propagent que localement, ce qui amène à des incohérences entre les voisins. D'autre part, la stratégie globale introduite par Horn et Schunck [30] est bien mieux adaptée à notre contexte. Bien que moins robuste au bruit que les méthodes locales telles que Lucas-Kanade, elle permet la propagation de contraintes éparses sur toute la surface. En outre, le modèle de déformation associé a prouvé son efficacité dans le domaine du graphisme [70].

L'extension du modèle de déformation d'Horn et Schunck à des points 3D est décrit par la fonction d'erreur suivante qui assure une rigidité locale du champ de mouvement :

$$\mathbf{E}_d = \int_S \|\nabla V\|^2 \mathrm{d}\mathbf{P}\,. \tag{6.4}$$

6.5.2 Minimisation de la fonctionnelle d'énergie

Nous trouvons le meilleur déplacement qui satisfait toutes les contraintes susmentionnées en minimisant la fonctionnelle d'erreur suivante :

$$\arg\min_V \left[\lambda_{3D}^2 \mathbf{E}_{3D} + \lambda_{2D}^2 \mathbf{E}_{2D} + \lambda_{flow}^2 \mathbf{E}_{flow} + \lambda_d^2 \mathbf{E}_d\right]\,, \tag{6.5}$$

où les différents coefficients λ sont des paramètres qui peuvent être configurés pour donner plus de poids à une contrainte particulière.

Cette fonctionnelle peut être minimisée en résolvant l'équation d'Euler-Lagrange qui lui est associée :

$$\begin{aligned}\sum_{c=1}^{N} \Bigg[\lambda_{flow}^2 \left[\nabla I_c^t \cdot \left[J_{\mathbf{\Pi}_c} V^t\right] + \frac{\mathrm{d}I_c^t}{\mathrm{d}t}\right] + \lambda_{2D}^2 \delta_{\Omega_{c,s}^t} J_{\mathbf{\Pi}_c}\left[V^t - V_{c,s}^t\right]\Bigg] \\ +\lambda_{3D}^2 \delta_{\mathcal{S}_m^t}\left[V^t - V_m^t\right] + \lambda_d^2 \nabla^2 V^t = 0\,,\end{aligned} \tag{6.6}$$

où δ est le symbole de Kronecher, qui précise que les contraintes éparses ne sont définies que pour certains points 3D appartenant à $\mathcal{S}_m^t$ ou leurs projections à $\Omega_{c,s}^t$.

L'équation d'Euler-Lagrange discrétisée pour chaque point 3D $\mathbf{P}$ de la surface a la forme :

$$\mathbf{A}_\mathbf{P} V_\mathbf{P} + \mathbf{b}_\mathbf{P} - \Delta V_\mathbf{P} = 0\,, \tag{6.7}$$

où Δ est l'opérateur de Laplace-Beltrami normalisé sur la surface.

La combinaison de l'équation (6.7) pour tous les points 3D $\mathbf{P} \in \mathcal{S}^t$ crée un système linéaire simple de la forme :

$$\begin{bmatrix} \mathbf{L} \\ \mathbf{A} \end{bmatrix} V^t + \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} = 0\,, \tag{6.8}$$

où $\mathbf{L}$ est la matrice laplacienne définie dans [70]. C'est un système linéaire très épars qui peut être résolu en utilisant n'importe quel *solver* éparse, *Taucs*, par exemple.

Il est intéressant de noter que cette formulation revisite la méthode d'édition Laplacienne de maillage *« as-rigid-as-possible »* de la communauté du graphisme [70]. Il est connu que ce modèle de déformation ne gère pas explicitement les rotations de la surface. Bien que cela soit un problème quand on déforme la surface avec peu de contraintes, ce qui est courant dans les applications de graphisme, la densité de contrainte de flot normal dans notre cas aide à récupérer les rotations sans pour autant passer par une optimisation non-linéaire du problème.

L'équation (6.6) peut aussi être résolue itérativement en utilisant la méthode de Jacobi appliquée à ce large système linéaire. Dans ce cas, il est possible de résoudre ce système linéaire pour chaque point indépendamment et de répéter cette étape itérativement en utilisant la solution mise à jour des points voisins. Cette approche variationnelle permet également d'affiner les résultats dans un contexte multi-échelle.

6.5.3 Sélection des poids et affinage itératif

Dans l'équation (6.6), les paramètres λ_{2D}, λ_{3D}, λ_{flow} et λ_d indiquent le poids, respectivement, des points d'intérêts 2D et 3D, du flot normal 2D et du Laplacien. Une forte valeur indique une influence plus importante pour le terme associé.

Dans notre contexte, de manière similaire à [81] en 2D, nous faisons confiance à nos points d'intérêts pour être robustes même lors de grands déplacements et nous sommes conscients que les contraintes de flot ne sont pas fiables quand la reprojection du déplacement est plus grande que quelques pixels dans les images. En conséquence, nous proposons une méthode itérative qui effectue deux minimisations successives de la fonctionnelle d'énergie avec deux jeux de paramètres différents. Les étapes de notre algorithme sont les suivantes :

1. Nous commençons par calculer les correspondances éparses 2D et 3D entre $\mathcal{S}^t$ et $\mathcal{S}^{t+1}$ et entre $\mathcal{I}^t$ et $\mathcal{I}^{t+1}$. Nous calculons également la matrice Laplacienne $\mathbf{L}$ de notre surface discrétisée.
2. Nous résolvons l'équation (6.8), avec $\lambda_{flow} = 0$ et des valeurs plus importantes pour λ_{3D} et λ_{2D} que pour λ_d. Nous obtenons alors une première estimation de V^t dénotée V'^t qui récupère les grands déplacements de la surface.

3. Nous créons une surface déformée $\mathcal{S}'^t = \mathcal{S}^t + V'^t$ que nous projetons dans toutes les caméras en utilisant l'information de texture d'origine, venant de la projection de $\mathcal{I}^t$ sur $\mathcal{S}^t$. Nous obtenons alors un nouveau jeu d'images $\mathcal{I}'^t$.

4. Nous calculons alors la visibilité de la surface S'^t sur chaque caméra ainsi que les contraintes denses de flot normal entre $\mathcal{I}'^t$ et $\mathcal{I}^{t+1}$ pour chaque point visible de la surface. Nous obtenons donc plusieurs contraintes par points échantillonnés sur la surface.

5. Tout comme dans l'étape 2, nous résolvons l'équation (6.8) en utilisant le flot calculé dans l'étape 4 et les points d'intérêts 2D et 3D calculés précédemment dans l'étape 1. Ces derniers sont utilisés comme des points d'ancrage ayant une contrainte de déplacement nul. Pour cette étape, nous utilisons des valeurs fortes de λ_{3D} et λ_{2D} et des valeurs plus faibles pour λ_{flow} et λ_d. Nous obtenons alors le déplacement entre $\mathcal{S}'^t$ et $\mathcal{S}^{t+1}$ dénoté V''^t et donc également une version raffinée de $V^t = V'^t + V''^t$. Cette seconde minimisation permet de récupérer de plus petits déplacements, mieux contraints par les contraintes de flot.

Nous avons observé par nos résultats que, dans la pratique, notre approche peut gérer aussi bien de grands déplacements que des petits. Ceci grâce aux points d'intérêts qui gèrent bien les grands déplacements et au flot de normal qui récupère mieux les détails précis.

6.6 Évaluation

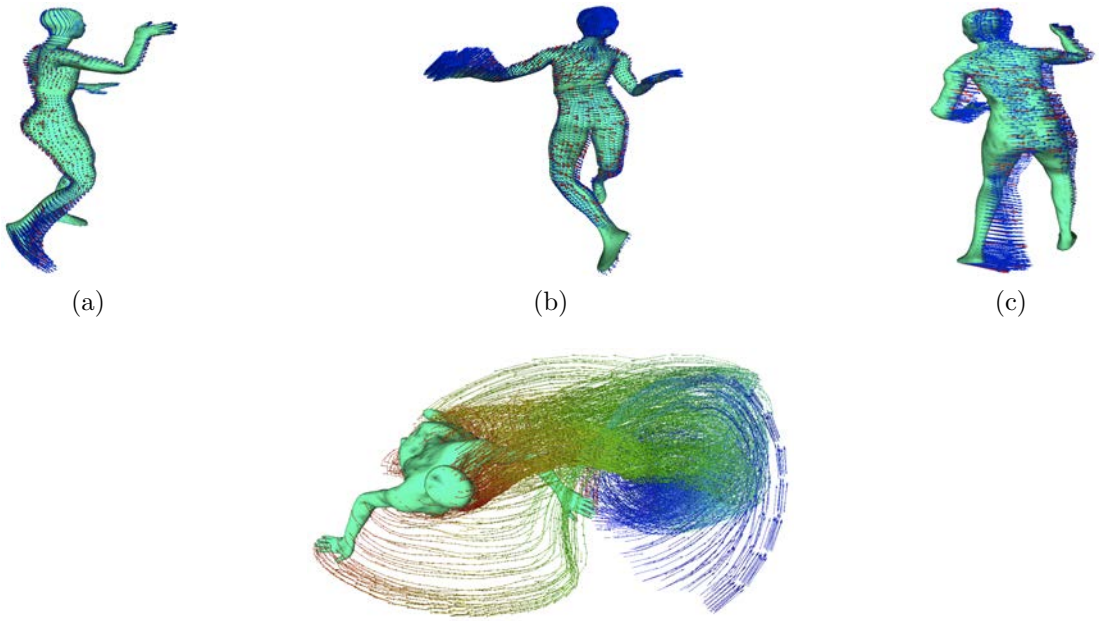
Pour notre évaluation nous avons utilisé aussi bien des données synthétiques que des données réelles :

1. Les données synthétiques ont été obtenues grâce à un modèle d'humain articulé, déformé au cours du temps pour créer une séquence de danse. Nous avons rendu cette séquence dans dix caméras virtuelles de résolution 1 MPixels, réparties sur une sphère autour de la danseuse. Le modèle utilisé est un maillage triangulaire avec $7K$ sommets, déformé pour générer une séquence de 200 trames.

2. Les données réelles ont été récupérées à partir de banques de données accessibles au public. La première séquence a été prise à partir de 32 caméras 2 MPixels dans la plateforme *GrImage*. Les maillages, obtenus avec EPVH, comportent $\sim 10K$ sommets. Nous avons également utilisé la séquence du *flashkick* de la base de données multi-vidéo *SurfCap* [72] de l'Université de Surrey. Cette séquence a été enregistrée à partir de huit caméras 2 MPixels, et produit des maillages lisses de $\sim 140K$ sommets.

6.6.1 Évaluation quantitative sur des données synthétiques

Grâce à l'algorithme décrit dans la section §6.5.3, nous avons calculé les champs de mouvement sur la séquence synthétique de la danseuse. La figure 6.4-(a)-(b)-(c) montre le champ de déplacement sur une des trames de la séquence. Les flèches rouges désignent les contraintes issues des points d'intérêts 3D et de la projection des points d'intérêts 2D, alors que les bleues désignent les vecteurs du champ de déplacement dense 3D.



(a) (b) (c)

(d)

FIGURE 6.4 – Champ de déplacement sur plusieurs trames de notre séquence synthétique de danseuse, et historique du mouvement sur plusieurs trames vu du dessus (les couleurs indiquent l'ancienneté du mouvement).

La figure 6.4-(d) montre le champ de déplacement accumulé sur plusieurs trames à partir d'une vue de dessus. Ce résultat peut être en quelque sorte comparé à celui de Varanasi et al. [76], en effet leur méthode, permet de mettre en correspondance deux maillages consécutifs dans le temps, et est capable de fournir un champ de vitesse en conséquence de cet appariement.

Comme les maillages sont cohérents dans le temps, nous avons pu obtenir la réalité terrain et donc évaluer nos résultats quantitativement. La figure 6.5 montre l'erreur sur l'angle et la taille de chaque vecteur de mouvement après chacune des deux étapes de régularisation de notre algorithme. Nous pouvons voir les avantages de l'utilisation des contraintes de flot normal pour affiner le champ de déplacement (voir gros plan sur la figure 6.6).

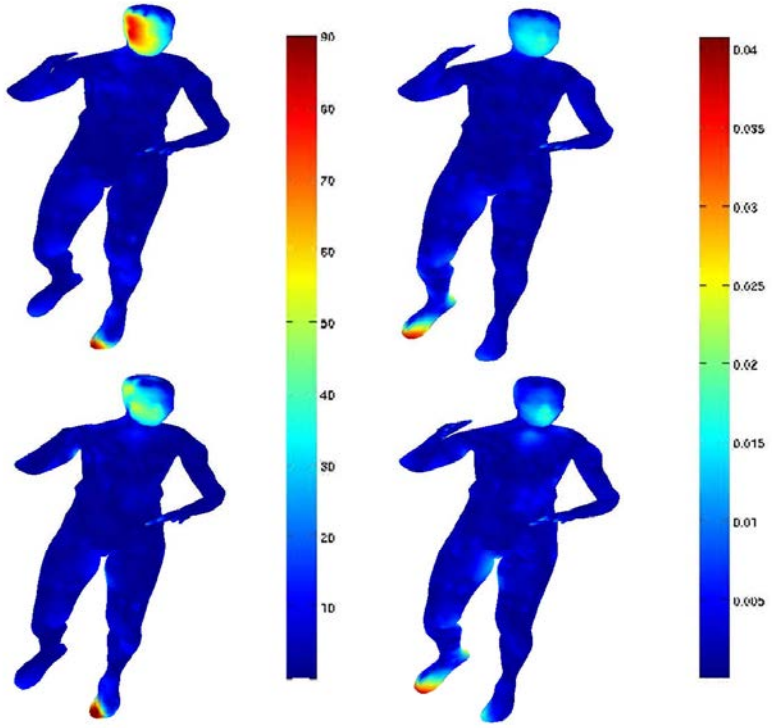


FIGURE 6.5 – Erreur sur le champs de déplacement : en angle en degré (gauche) et en norme en mètre (droite), après la première (haut) et la deuxième (bas) régularisation.

(a) Première régularisation (b) Seconde régularisation

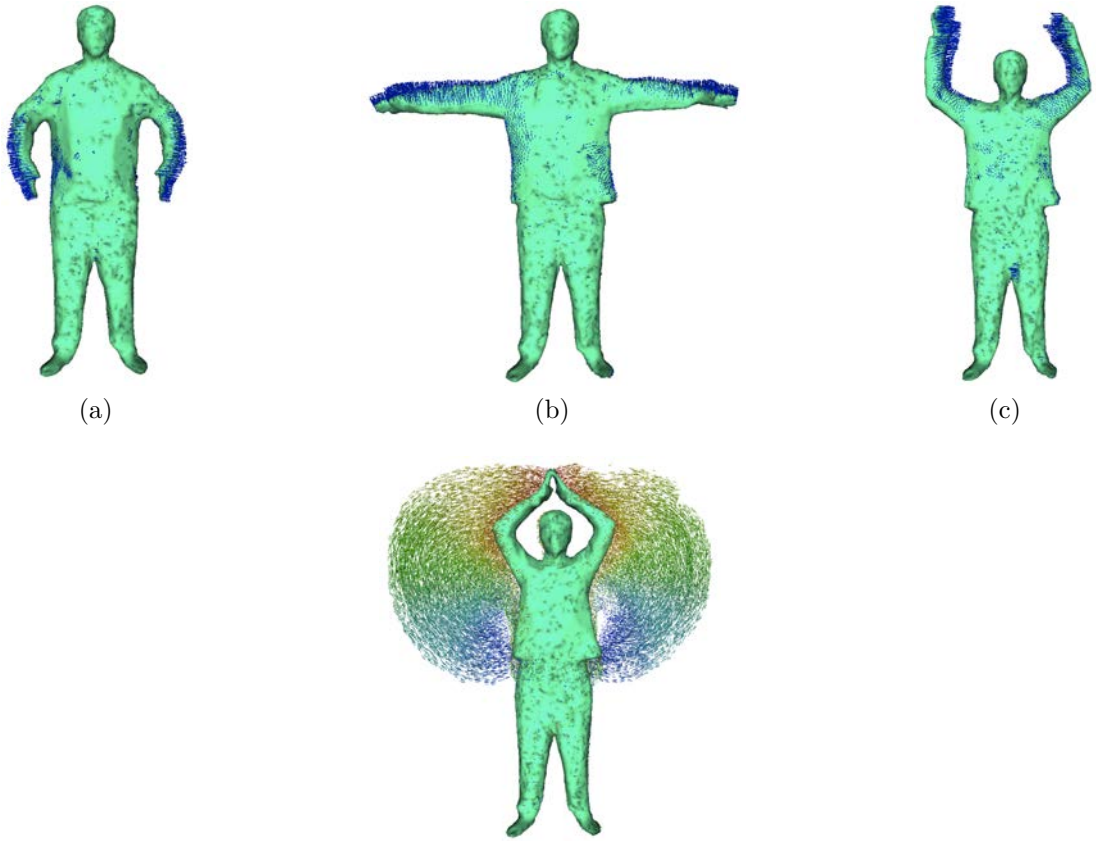
FIGURE 6.6 – Gros plan sur l'erreur en angle sur la face de la danseuse. Ces images montrent l'amélioration après la deuxième étape de régularisation qui aide pour récupérer les petits déplacements.

6.6.2 Expériences sur des données réelles

Notre première séquence réelle montre un sujet qui réalise des actions simples : il déplace ses deux mains à partir des hanches jusqu'au dessus de sa tête. Le sujet

porte des vêtements amples et bien texturés ce qui permet de calculer un nombre élevé et fiable de correspondances 2D et 3D.

Les figures 6.7-(a)-(b)-(c) montrent le mouvement instantané récupéré en utilisant notre méthode. Notez que nous ne calculons qu'un mouvement dense sur la surface et non une déformation du maillage. Ainsi, nous n'avons pas une connectivité constante dans le temps et nous ne pouvons pas effectuer le suivi des sommets du maillage sur toute la séquence. Par conséquent, l'évaluation quantitative des données n'est pas possible, mais la visualisation des résultats est très satisfaisante. La figure 6.7-(d) montre le champ de déplacement accumulé sur toute la séquence.



(a) (b) (c)

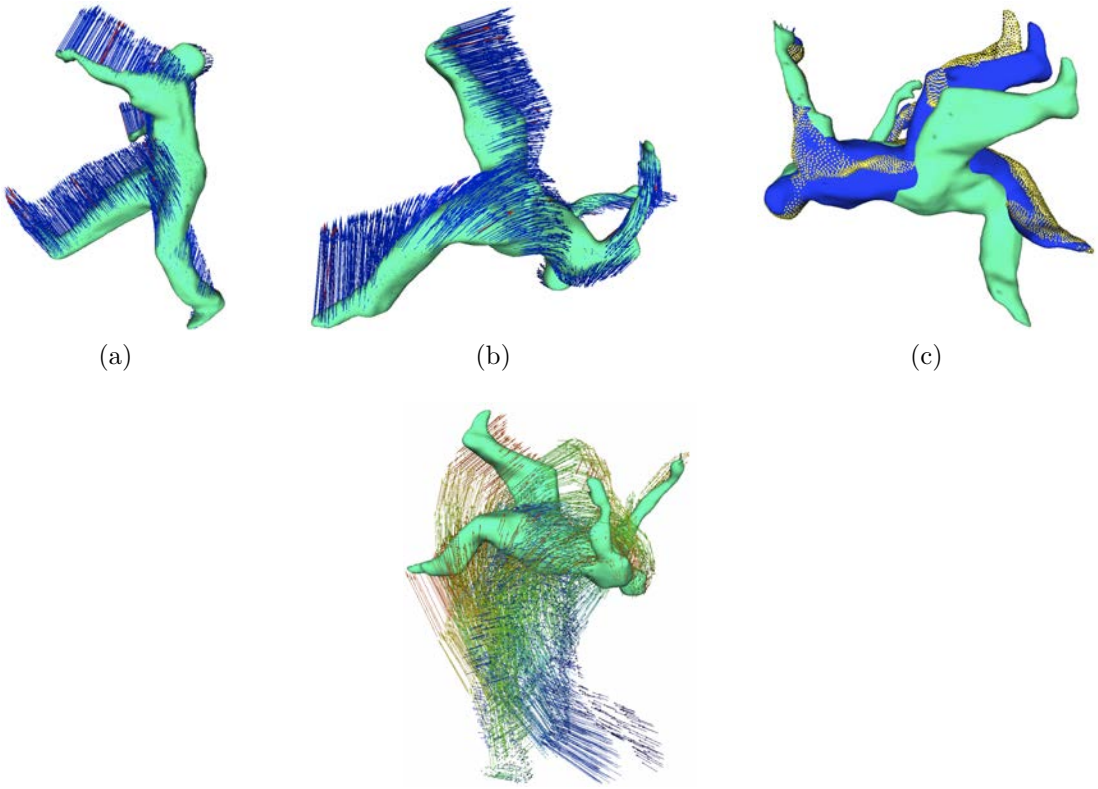
(d)

FIGURE 6.7 – Champs de déplacement sur certaines trames de nos données réelles, et historique du mouvement accumulé sur toute la séquence (les couleurs indiquent l'ancienneté du mouvement).

Nous avons également calculé le champ de déplacement 3D sur la séquence du *flashkick* qui est très populaire. Dans cette séquence difficile, le sujet porte des vêtements amples avec peu d'information de texture. En outre, l'amplitude du mouvement est très élevée entre deux trames. Nous pouvons donc calculer moins

de correspondances 2D et 3D. Elles sont pourtant nécessaires pour récupérer les gros déplacements.

Nous avons cependant réussi à calculer un champ de mouvement cohérent sur la plupart des trames (voir les figures 6.8-(a)-(b)). Sur certaines trames, notre algorithme n'a pas trouvé de points d'intérêts sur les jambes ou les pieds du danseur, le champ de mouvement calculé à partir de ces indices montre bien la bonne direction, mais pas la bonne norme des vecteurs. Le manque de contraintes visuelles pour la première estimation du champ de mouvement ne permet pas de calculer certains déplacements complètement, le déplacement restant ne peut pas être récupéré entièrement avec les contraintes de flot normal. La figure 6.8-(c) montre une trame problématique où le mouvement de la jambe droite du danseur n'est pas correctement calculé. Pour visualiser cette erreur, nous avons affiché les surfaces d'entrée au temps t et $t+1$ (respectivement cyan et bleu foncé), tandis que la surface déplacée avec le champ de mouvement calculé est indiquée par des points jaunes. Enfin, la figure 6.8-(d) montre l'historique du mouvement sur quelques trames.



(a) (b) (c)

(d)

FIGURE 6.8 – Champs de déplacement sur plusieurs trames de la séquence du flash-kick (a) et (b), mouvement partiellement calculé (c), et historique du mouvement sur toute la séquence (d) (les couleurs indiquent l'ancienneté).

La vidéo 4 montre plus en détails les résultats obtenus.

6.7 Bilan

La contribution de cette partie est double : premièrement, nous avons présenté une méthode unifiée qui permet de combiner des informations photométriques pour calculer le mouvement d'une surface, d'autre part, nous avons introduit une méthode itérative qui permet de gérer de gros déplacements tout en récupérant les petits détails.

Comme le montrent les résultats, notre méthode est assez robuste. Néanmoins, nos expériences ont mis en évidence certaines faiblesses potentielles. Comme nous le pensions, en s'appuyer sur des caractéristiques visuelles impose d'avoir une bonne information de texture dans les images. Notre méthode pourrait être améliorée par l'ajout d'autres contraintes, par exemple, un critère de cohérence photométrique tel que celui utilisé par Pons et al. dans [61].

La méthode que nous proposons donne de toute façon des informations utiles et fiables sur les propriétés intrinsèques d'une séquence 4D. La connaissance du déplacement instantané peut être utilisée comme donnée d'entrée pour de nombreuses tâches en vision par ordinateur, telle que le suivi de surface, le transfert de mouvement ou la segmentation de maillage.

Même si nous n'avons pas mis l'accent sur les performances de calcul pour notre première implémentation, nous sommes certains que la plupart des calculs pourraient s'exécuter en parallèle. En effet, l'extraction des points d'intérêts 2D, ainsi que le calcul des contraintes de flot normal, sont indépendants par caméra. Fe plus, des implémentations temps-réel de SIFT et des méthodes de flux optique existent déjà. La propriété linéaire de la régularisation permet de s'attendre à une exécution en temps-réel également.

Pour le moment, les systèmes multi-caméra qui, comme le système *GrImage*, permettent de faire de la reconstruction 3D en temps-réel ne calculent pas vraiment le mouvement associé au modèle. Ces informations supplémentaires pourraient améliorer considérablement les applications interactives telles que les interactions basées sur des collisions entre le sujet reconstruit et toutes sortes d'objets virtuels. Il pourrait également être utilisé pour la reconnaissance d'action. La prochaine étape dans notre cas est d'intégrer cette méthode au pipeline de reconstruction 3D de la plateforme *GrImage*.

Conclusion

« Wake up Neo
The Matrix has you...
Follow the white rabbit...
Knock knock Neo. »
Matrix

Chapitre 7

Conclusions et perspectives

Pendant ma thèse, nous nous sommes intéressés à l'utilisation des systèmes de vision comme périphérique d'entrée dans les environnements virtuels, dans le but d'améliorer la présence des utilisateurs, leur sentiment d'immersion et leurs possibilités d'interaction. Nous avons montré l'efficacité et l'intérêt de ces systèmes grâce à plusieurs contributions :

1. Nous avons montré qu'il est possible d'utiliser plusieurs systèmes de reconstruction 3D temps-réel dans un contexte partagé et d'envoyer la représentation 3D de chaque utilisateur à d'autres utilisateurs, sur le réseau, à un taux de rafraîchissement interactif. Cette modélisation 3D peut permettre aux utilisateurs d'être présents virtuellement au travers de leur clone numérique, et d'avoir le sentiment que les autres utilisateurs sont également présents. Ce clone numérique a des caractéristiques intrinsèques intéressantes, son apparence est semblable à celle de l'utilisateur et il permet de réaliser des interactions physiques avec le monde virtuel sans pour autant avoir à s'équiper de périphériques ou marqueurs encombrants. Il permet également de communiquer grâce aux expressions faciales ou aux gestes, et ceci de la manière la plus naturelle qui soit.

   Pour arriver à cela, nous avons mis en place une application qui permet à plusieurs utilisateurs, situés à quelques centaines de kilomètres les uns des autres, de se rencontrer, en 3D, dans un environnement virtuel partagé, et de collaborer en interagissant ensemble avec le monde virtuel.

2. Dans le but de mixer correctement le monde réel avec le monde virtuel et d'immerger complètement l'utilisateur dans l'environnement virtuel, nous avons couplé notre plateforme de reconstruction 3D multi-caméra avec un système de visualisation immersive unidirectionnelle. Pour cela, nous avons intégré un casque de réalité virtuelle au sein de la plateforme. Cette intégration donne le sentiment à l'utilisateur de ne faire plus qu'un avec son clone virtuel.

Les applications de ce concept donnent la possibilité aux utilisateurs de voir le modèle 3D de leur clone numérique aligné de manière exacte avec leur propre corps et correctement mixé avec les objets virtuels. En utilisant un moteur de simulation physique, l'utilisateur a la possibilité d'interagir avec les objets virtuels, à la première personne, et de façon co-localisée. Avec ces expérimentations, nous avons démontré l'intérêt du couplage d'une plateforme de reconstruction 3D temps-réel avec un système de visualisation à la première personne.

Nous avons également ajouté une caméra mobile placée sur le *HMD* pour améliorer la qualité de la reconstruction 3D et des données photométriques dans le champs de vision de l'utilisateur. Les premières expérimentations ont été effectuées et ont montré l'intérêt du concept et le gain en termes de qualité. Une utilisation de cette technique dans un cadre applicatif reste tout de même problématique, car assujettie à la précision de la technologie utilisée pour suivre la tête de l'utilisateur. Le matériel que nous avons à notre disposition ne nous a donc pas permis de tester ce concept de façon pérenne dans des applications interactives.

3. Dans une volonté de permettre des interactions plus réalistes entre l'utilisateur reconstruit en 3D et l'environnement virtuel, nous avons voulu augmenter la quantité d'informations disponible sur l'utilisateur. Pour cela, nous avons développé une méthode qui permet de récupérer un champs de vitesse 3D dense à la surface du modèle reconstruit. Cette méthode se base sur les informations de flot optique dans les images issues du système multi-caméra et sur la mise en correspondance de points d'intérêts 2D et 3D.

   Bien que le temps nous ait manqué pour étendre cette méthode à un contexte temps-réel, nous sommes sûrs qu'une implémentation distribuée peut être envisagée. Les champs de vitesse 3D ainsi calculés pourront être utilisés comme information bas niveau pour gérer de façon plus réaliste les collisions au sein du moteur de simulation physique, ou pour développer des méthodes de suivi de maillage, de segmentation et reconnaissance de formes ou, encore, pour créer de nouveaux paradigmes d'interaction.

Ces différentes contributions montrent les possibilités offertes par les systèmes multi-caméra dans le domaine de la réalité virtuelle. Il est possible d'immerger un utilisateur au sein d'un environnement virtuel partagé, de lui donner le sentiment d'être présent dans cet environnement, physiquement et mécaniquement, et de donner aux autres utilisateurs le sentiment d'être en face d'une vraie personne et non d'un avatar.

Les concepts introduits et les expérimentations effectuées sont la base de ce qui pourrait être utilisé dans le futur, un nouveau mode de communication et de collaboration à distance. Il est en effet évident que le sentiment de présence de l'utilisateur passe par son identification à l'avatar qui le représente dans le monde virtuel. Notre système permet de donner à cet avatar l'apparence même de l'utilisateur, de véhiculer ses expressions faciales et ses gestes, et une interaction naturelle. Cela facilite ce sentiment d'identification.

Il reste bien sur de nombreuses voies à explorer dans tous les domaines étudiés pendant ma thèse :

Vision par ordinateur : En termes d'évolution au niveau du système de reconstruction 3D multi-caméra lui-même, les efforts doivent se concentrer suivant ces points :

- Une solution de soustraction de fond qui permettrait d'avoir un fond dynamique doit être développée. Cela faciliterait l'installation du système dans n'importe quelle pièce, voire même en extérieur. Des pistes existent déjà :
  - Il est possible, par exemple, d'utiliser la redondance des données dans les différentes caméras du système pour créer une solution de soustraction de fond multi-caméra indépendante d'un modèle du fond.
  - L'utilisation de caméra temps-de-vol couplée à notre système pourrait également aider à segmenter le fond.
  - Enfin, les méthodes probabilistes basées sur les grilles d'occupation constituent également une piste à explorer.

  La nécessité de segmenter le fond en temps-réel avec des latences minimum reste la contrainte majeur à toute implémentation.

- L'agrandissement de l'espace d'acquisition est aussi un point important pour rendre le système plus aisément utilisable. Ceci implique de multiplier le nombre de caméras et de développer des méthodes de reconstruction 3D à partir de sous-ensembles de caméras et de les fusionner. Ceci n'est pas encore possible avec la version actuelle d'EPVH.

- Le modèle produit par EPVH, complété par les textures issues des caméras d'origine, permet d'obtenir une visualisation assez fidèle de l'utilisateur. L'absence des concavités peut poser des problèmes lors de l'utilisation de ce modèle. Le couplage avec d'autres techniques de reconstruction 3D pour *creuser* l'enveloppe visuelle produite par EPVH devrait permettre d'améliorer aussi bien le rendu final que les interactions possibles avec le modèle.

Système : Du point de vue du système, les efforts doivent être essentiellement concentrés dans l'accélération des différentes étapes du pipeline de reconstruction

3D. Ceci dans le but de diminuer la latence de la reconstruction 3D et donc d'améliorer l'interactivité :

- En profitant de la multiplication du nombre de coeurs accessibles dans les machines récentes et des processeurs graphiques très performants, les algorithmes décrits dans cette thèse pourraient être grandement accélérés. L'application pourrait aussi profiter des mécanismes d'adaptation automatique de la distribution des processus en fonction de la charge.

- Les processus de ré-échantillonnage des données peuvent également être améliorés pour s'adapter à la charge du système ou à une situation physique changeante (arrivée d'une deuxième personne dans la scène, par exemple). Pour cela ils pourraient jouer sur des paramètres de l'application, comme le niveau de discrétisation des silhouettes ou la réduction de la taille des images, pour conserver de l'interactivité sans diminuer la fréquence ou augmenter la latence de la reconstruction 3D. Il peut être également envisageable de contrôler directement la fréquence des caméras en fonction de la charge du système afin d'éviter de traiter des trames qui seront supprimées par la suite.
- L'intergiciel *FlowVR* pourrait aussi être amélioré pour permettre le lancement dynamique des modules d'une application afin d'offrir plus de flexibilité à l'exécution.

Téléprésence et réseau : Afin d'améliorer les performances des applications interactives à distance un effort devrait être fait sur le transfert des données 3D :

- Des algorithmes de compression de maillages et de textures devront être développés pour permettre aux utilisateurs d'envoyer, à des fréquences interactives, et sous des contraintes de bande passante plus réduite, un maillage et ses textures associées, à plusieurs utilisateurs distants. L'utilisation de la prédiction du maillage ou des textures, basée sur les champs de vitesse 3D, étudiés dans cette thèse, pourrait être une solution pour limiter la taille des données à envoyer sur le réseau.

- Des algorithmes de suivi de maillage ou de reconnaissance de squelette peuvent aussi être utilisés pour réduire les données à envoyer sur le réseau. Il pourrait être envisagé de n'envoyer qu'un maillage et un atlas de texture à intervalles réguliers puis de n'envoyer que des modifications de la structure entre ces *trames clés*. Ce principe, utilisé pour l'encodage des vidéos, pourrait donc être étendu à la 3D.

Immersion et *HMD* : Du côté de l'immersion des utilisateurs, la solution proposée dans cette thèse, à savoir l'utilisation d'un *HMD* pour permettre une

visualisation à la première personne et des interactions co-localisées, est très satisfaisante.

Elle présente quand même un défaut de taille, à savoir que le casque couvre une partie importante de la tête de l'utilisateur, cruciale pour la communication à cause des expressions faciales. Des solutions peuvent être envisagées pour remplacer le modèle 3D du casque par un modèle pré-enregistré de la tête de l'utilisateur.

Des expérimentations doivent aussi être effectuées en utilisant un casque plus immersif. Ce dernier devrait avoir un champs de vision et une résolution maximal et pouvoir afficher des images en relief grâce à la stéréoscopie.

L'amélioration de la qualité de la reconstruction 3D et des textures plaquées sur le modèle de l'utilisateur devrait pouvoir être envisagée grâce à un traqueur de meilleure qualité et une caméra montée sur le *HMD*. Ceci reste tout de même à tester car il n'est pas dit qu'un seul gain en précision dans le suivi des cibles soit suffisant pour améliorer de façon efficace le calcul de la position de la caméra sur le casque. Il pourrait être plus intéressant d'utiliser les caméras du système pour suivre la cible du casque parallèlement à l'étape de reconstruction 3D et obtenir un positionnement de la caméra plus précis.

Interaction réel/virtuel : Bien que les possibilités d'interaction soient déjà convaincantes, il reste des avancées à faire dans ce domaine. De nouvelles méthodes doivent être développées pour permettre d'identifier et de suivre des parties du corps de l'utilisateur, et de nouveaux paradigmes d'interaction profitant de toutes ces informations sur l'utilisateur doivent être imaginés.

Enfin, afin de compléter la boucle d'interaction homme-machine, il faudrait ajouter à notre système des possibilités de retour haptique. L'utilisateur pourrait alors sentir le contact des objets virtuels, et, pourquoi pas, des autres utilisateurs. La technologie actuelle ne permet pas de retour haptique sans utilisation de périphériques encombrants, mais peut-être que de futurs technologies répondant à ce besoin. Il faudra néanmoins résoudre le problème de l'identification et du suivi des différentes parties du corps de l'utilisateur afin de pouvoir appliquer une force réelle en fonction d'une collision virtuelle. Il faut pour cela une correspondance entre le maillage modélisé en 3D et le corps de l'utilisateur.

La combinaison de toutes ces améliorations devraient permettre la création d'une plateforme de virtualisation où l'utilisateur serait complètement immergé, évoluerait dans un monde virtuel aussi naturellement que dans la réalité et collaborerait avec d'autres utilisateurs situés à l'autre bout de la planète. Nous ne sommes donc pas loin de la création d'un *Metaverse* accessible à tous, à l'image de celui imaginé par Neal Stephenson.

Bibliographie

[1] J. Allard, E. Boyer, J.-S. Franco, C. Ménier et B. Raffin : Marker-Less Real Time 3D Modeling for Virtual Reality. *In Immersive Projection Technology*, 2004. 5, 26

[2] J. Allard, V. Gouranton, L. Lecointre, S. Limet, E. Melin, B. Raffin et S. Robert : FlowVR : a Middleware for Large Scale Virtual Reality Applications. *In Euro-par*, 2004. 39

[3] J. Allard, C. Ménier, B. Raffin, E. Boyer et F. Faure : GrImage : Markerless 3D Interactions. *In ACM Sigggraph - Emerging Technologies*, 2007. 5, 9, 26, 37, 51

[4] J. Allard et B. Raffin : A Shader-Based Parallel Rendering Framework. *In IEEE Visualization*, 2005. 38, 55

[5] W. S. Bainbridge : The Scientific Research Potential of Virtual Worlds. *Science*, 2007. 14

[6] J.L. Barron, D.J. Fleet et S.S. Beauchemin : Performance of Optical Flow Techniques. *International Journal of Computer Vision*, 1994. 103, 110

[7] E. Boyer : On Using Silhouettes for Camera Calibration. *In Asian Conference on Computer Vision*, 2006. 29

[8] C. Cagniart, E. Boyer et S. Ilic : Probabilistic Deformable Surface Tracking From Multiple Videos. *In European Conference on Computer Vision*, 2010. 105

[9] J. Carranza, C. Theobalt, M.A. Magnor et H.-P. Seidel : Free-Viewpoint Video of Human Actors. *ACM Siggraph*, 2003. 25

[10] E. Chen et L. Williams : View Interpolation for Image Synthesis. *In ACM Siggraph*, 1993. 23

[11] K. M. Cheung, T. Kanade, J.-Y. Bouguet et M. Holler : A Real Time System for Robust 3D Voxel Reconstruction of Human Motions. *In IEEE Conference on Computer Vision and Pattern Recognition*, 2000. 24, 25, 30

[12] K. Cleland : Image Avatars and Media Mirrors. *In Engage : Audience, Art and Interaction*, 2006. 83

[13] J. Crowley, F. Berard et J. Coutaz : Finger Tracking as an Input Device for Augmented Reality. *In International Workshop on Automatic Face and Gesture Recognition*, 1995. 20

[14] C. Cruz-Neira, D. J. Sandin et T. A. DeFanti : Surround-Screen Projection-based Virtual Reality : The Design and Implementation of the CAVE. *In ACM Siggraph*, 1993. 15

[15] T. Darrel et A.P. Pentland : Attention-driven Expression and Gesture Analysis in an Interactive Environment. *In International Workshop on Automatic Face and Gesture Recognition*, 1995. 20

[16] I. Debled-Rennesson, S. Tabbone et L. Wendling : Fast Polygonal Approximation of Digital Curves. *International Conference on Pattern Recognition*, 2004. 31

[17] D.F. DeMenthon et L.S. Davis : Model-based Object Pose in 25 Lines of Code. *International Journal of Computer Vision*, 1995. 74

[18] F. Devernay, D. Mateus et M. Guilbert : Multi-Camera Scene Flow by Tracking 3-D Points and Surfels. *In Computer Vision and Pattern Recognition*, 2006. 111

[19] J.-S. Franco et E. Boyer : Exact Polyhedral Visual Hulls. *In British Machine Vision Conference*, 2003. 32

[20] J.-S. Franco et E. Boyer : Efficient Polyhedral Modeling from Silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008. 32, 33

[21] J.-S. Franco, C. Ménier, E. Boyer et B. Raffin : A Distributed Approach for Real Time 3D Modeling. *In Conference on Computer Vision and Pattern Recognition Workshop*, 2004. 36

[22] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn et H.-P. Seidel : Motion Capture Using Joint Skeleton Tracking and Surface Estimation. *In International Conference on Computer Vision and Pattern Recognition*, 2009. 31

[23] W. Gibson : *Neuromancer*. Ace, 1984. 3

[24] S.J. Gilson, A.W. Fitzgibbon et A. Glennerster : An Automated Calibration Method for Non-See-Through Head Mounted Displays. *Computer Graphics and Virtual Reality*, 2009. 75, 82

[25] O. Grau, T. Pullen et G.A. Thomas : A Combined Studio Production System for 3D Capturing of Live Action and Immersive Actor Feedback. *IEEE Transactions on Circuits and Systems for Video Technology*, 2004. 23

[26] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L.V. Gool, S. Lang, K. Strehlke, A. V. Moere et O. Staadt : Blue-C : A Spatially Immersive Display and 3D Video Portal for Telepresence. *ACM Transactions on Graphics*, 2003. 46, 71

[27] M. Hachet, F. Dècle, S. Knödel et P. Guitton : Navidget for 3D Interaction : Camera Positioning and Further Uses. *International Journal of Human-Computer Studies*, 2009. 58

[28] J.-M. Hasenfratz, M. Lapierre et F. Sillion : A Real-time System for Full Body Interaction with Virtual Worlds. *Eurographics Symposium on Virtual Environments*, 2004. 25

[29] A. Hilton et J. Starck : Multiple View Reconstruction of People. *In International Symposium on 3D Data Processing, Visualization and Transmission*, 2004. 23

[30] B.K.P. Horn et B.G. Schunck : Determining Optical Flow. *Artificial Intelligence*, 1981. 103, 111

[31] T. Horprasert, D. Harwood et L. S. Davis : A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection. *In IEEE International Conference on Computer Vision*, 1999. 30

[32] M. Isard et J. MacCormick : Dense Motion and Disparity Estimation via Loopy Belief Propagation. *In Asian Conference on Computer Vision*, 2006. 105

[33] T. Kanade, P. Rander et P.J. Narayanan : Virtualized Reality : Constructing Virtual Worlds from Real Scenes. *IEEE Multimedia, Immersive Telepresence*, 1997. 23

[34] P. Kauff et O. Schreer : An Immersive 3D Video-Conferencing System Using Shared Virtual Team User Environments. *In International Conference on Collaborative Virtual Environments*, 2002. 45

[35] A. Kolb, E. Barth, R. Koch et R. Larsen : Time-of-Flight Sensors in Computer Graphics. *In Eurographics 2009 - State of the Art Reports*, 2009. 21

[36] M.W. Krueger, T. Gionfriddo et K. Hinrichsen : Videoplace – an Artificial Reality. *In ACM SIGCHI*, 1985. 19

[37] S.A. Kuhl, W.B. Thompson et S.H. Creem-Regehr : HMD Calibration and its Effects on Distance Judgments. *ACM Transactions on Applied Perception*, 2009. 82

[38] G. Kurillo, R. Bajcsy, K. Nahrstedt et O. Kreylos : Immersive 3D Environment for Remote Collaboration and Training of Physical Activities. *In IEEE Virtual Reality*, 2008. 46

[39] A. Laurentini : The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1994. 24, 32

[40] S. Lazebnik, E. Boyer et J. Ponce : On How to Compute Exact Visual Hulls of Object Bounded by Smooth Surfaces. *In IEEE Conference on Computer Vision and Pattern Recognition*, 2001. 32

[41] J.-D. Lesage et B. Raffin : A Hierarchical Component Model for Large Parallel Interactive Applications. *Journal of Supercomputing*, 2008. 39

[42] J.-D. Lesage et B. Raffin : High Performance Interactive Computing with FlowVR. *In IEEE Virtual Reality SEARIS Workshop*, 2008. 39

[43] C. Liang et K.-Y. K. Wong : Exact Visual Hull from Marching Cubes. *In International Conference on Computer Vision Theory and Applications*, 2008. 25

[44] C. Liu, J. Yuen, A. Torralba, J. Sivic et W. Freeman : SIFT Flow : Dense Correspondence across Different Scenes. *In European Conference on Computer Vision*, 2008. 106

[45] D.G. Lowe : Distinctive Image Features from Scale-invariant Keypoints. *International Journal of Computer Vision*, 2004. 108

[46] B.D. Lucas et T. Kanade : An Iterative Image Registration Technique with an Application to Stereo Vision. *In International Joint Conference on Artificial Intelligence*, 1981. 103

[47] W. Matusik et H. Pfister : 3D TV : A Scalable System for Real-Time Acquisition, Transmission, and Autostereoscopic Display of Dynamic Scenes. *In ACM Siggraph*, 2004. 45

[48] M. McGinity, J. Shaw, V. Kuchelmeister, A. Hardjono et D.D. Favero : AVIE : a Versatile Multi-user Stereo 360° Interactive VR Theatre. *In Workshop on Emerging Displays Technologies*, 2007. 25

[49] C. Ménier : *Système de Vision Temps-réel pour les Interactions*. Thèse de doctorat, Institut National Polytechnique de Grenoble, 2007. 5, 36

[50] Brice Michoud, Erwan Guillou, Hector Briceno Pulido et Saida Bouakaz : Real-Time Marker-free Motion Capture from Multiple Cameras. *In International Conference on Computer Vision*, 2007. 25

[51] J. Mitchelson et A. Hilton : Wand-based Multiple Camera Studio Calibration. Rapport technique, CVSSP, 2003. 30

[52] J. Mulligan et K. Daniilidis : Real Time Trinocular Stereo for Tele-Immersion. *In International Conference on Image Processing*, 2001. 45

[53] A. Naveed, C. Theobalt, C. Rossl, S. Thurn et H.P. Seidel : Dense Correspondence Finding for Parametrization-free Animation Reconstruction from Video. *In Computer Vision and Pattern Recognition*, 2008. 105

[54] J. Neumann et Y. Aloimonos : Spatio-Temporal Stereo Using Multi-Resolution Subdivision Surfaces. *International Journal of Computer Vision*, 2002. 103, 105

[55] R. O'Hagan et A. Zelinsky : Finger Track - a Robust and Real-time Gesture Interface. *In Australian Joint Conference on Artificial Intelligence*, 1997. 20

[56] B. Petit, T. Dupeux, B. Bossavit, J. Legaux, B. Raffin, E. Melin, J.-S. Franco, I. Assenmacher et E. Boyer : A 3D Data Intensive Tele-immersive Grid. *In ACM Multimedia*, 2010. 7, 8, 63

[57] B. Petit, J.-D. Lesage, E. Boyer, J.-S. Franco et B. Raffin : Remote and Collaborative 3D Interactions. *In IEEE 3DTV Conference*. IEEE, 2009. 7, 8, 85

[58] B. Petit, J.-D. Lesage, J.-S. Franco, E. Boyer et B. Raffin : Grimage : 3d modeling for remote collaboration and telepresence. *In ACM Symposium on Virtual Reality Software and Technology*, 2008. 7, 8, 54

[59] B. Petit, J.-D. Lesage, C. Ménier, J. Allard, J.-S. Franco, B. Raffin, E. Boyer et F. Faure : Multi-Camera Real-Time 3D Modeling for Telepresence and Remote Collaboration. *International Journal of Digital Multimedia Broadcasting*, 2010. 7, 8, 26

[60] B. Petit, J.-D. Jean-Denis Lesage, E. Boyer et B. Raffin : Virtualization Gate. *In ACM Sigggraph - Emerging Technologies*, 2009. 8

[61] J.-P. Pons, R. Keriven et O. Faugeras : Modelling Dynamic Scenes by Registering Multi-View Image Sequences. *In Computer Vision and Pattern Recognition*, 2005. 105, 119

[62] Andreas Pusch : *Conflits visuo-proprioceptifs de la main pour l'interaction 3D en réalité augmentée*. Thèse de doctorat, INPG, 2008. 78

[63] D. Scharstein et R. Szeliski : A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 2002. 21

[64] M.J. Schuemie, P. van der Straaten, M. Krijn et C.A. van der Mast : Research on Presence in Virtual Reality : A Survey. *Cyberpsychology and Behavior*, 2001. 14

[65] S. Seitz, B. Curless, J. Diebel, D. Scharstein et R. Szeliski : A Comparison and Evaluation of Multi-view Stereo Reconstruction Algorithms. *In Conference on Computer Vision and Pattern Recognition*, 2006. 24, 31

[66] Z. Shujun, W. Cong, S. Xuqiang et W. Wei : DreamWorld : CUDA-Accelerated Real-time 3D Modeling System. *In IEEE VECIMS*, 2009. 25, 46

[67] S.N. Sinha et M. Pollefeys : Synchronization and Calibration of Camera Networks from Silhouettes. *In International Conference on Pattern Recognition*, 2004. 29

[68] G. Slabaugh, B. Culbertson, T. Malzbender et R. Schafe : A Survey of Methods for Volumetric Scene Reconstruction from Photographs. 24

[69] L. Soares, C. Ménier, B. Raffin et J.-L. Roch : Work Stealing for Time-constrained Octree Exploration : Application to Real-time 3D Modeling. *In Eurographics Symposium on Parallel Graphics and Visualization*, 2008. 25

[70] O. Sorkine et M. Alexa : As-Rigid-As-Possible Surface Modeling. *In Eurographics Symposium on Geometry Processing*, 2007. 111, 113

[71] J. Starck et A. Hilton : Correspondence Labeling for Wide-Timeframe Free-Form Surface Matching. *In European Conference on Computer Vision*, 2007. 108

[72] J. Starck et A. Hilton : Surface Capture for Performance-Based Animation. *IEEE Computer Graphics and Applications*, 2007. 105, 114

[73] N. Stephenson : *Snow Crash*. Bantam Books, 1992. 3

[74] R. Szeliski : Rapid Octree Construction from Image Sequences. *Computer Vision, Graphics and Image Processing*, 1993. 24

[75] M. Takemura et Y. Ohta : Diminishing Head-Mounted Display for Shared Mixed Reality. *In International Symposium on Mixed and Augmented Reality*, 2002. 88

[76] K. Varanasi, A. Zaharescu, E. Boyer et R.P. Horaud : Temporal Surface Tracking Using Mesh Evolution. *In European Conference on Computer Vision*, 2008. 105, 115

[77] S. Vedula, S. Baker, P. Rander, R. Collins et T. Kanade : Three-Dimensional Scene Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005. 103, 105, 110, 111

[78] D. Vlasic, I. Baran, W. Matusik et J. Popović : Articulated Mesh Animation from Multi-view Silhouettes. *ACM Transactions on Graphics*, 2008. 31

[79] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke et D. Cremeres : Efficient Dense Scene Flow from Sparse or Dense Stereo Data. *In European Conference on Computer Vision*, 2008. 105

[80] W. Wu, Z. Yang, D. Jin et K. Nahrstedt : Implementing a Distributed Tele-immersive System. *In IEEE International Symposium on Multimedia*, 2008. 45

[81] L. Xu, J. Jia et Y Matsushita : Motion Detail Preserving Optical Flow Estimation. *In Computer Vision and Pattern Recognition*, 2010. 103, 106, 113

[82] R. Yang et Z. Zhang : Eye Gaze Correction with Stereovision for Video-Teleconferencing. *In European Conference on Computer Vision*, 2002. 21

[83] A. Zaharescu, E. Boyer, K. Varanasi et R.P. Horaud : Surface Feature Detection and Description with Applications to Mesh Matching. *In IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 108

[84] A. Zelinsky et J. Heinzmann : Real-time Visual Recognition of Facial Gestures for Human Computer Interaction. *In International Conference on Automatic Face and Gesture Recognition*, 1996. 20

[85] Y. Zhang et C. Kambhamettu : On 3D Scene Flow and Structure Estimation. *In Computer Vision and Pattern Recognition*, 2001. 105

[86] Z. Zhang : Camera Calibration With One-Dimensional Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. 30

[87] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder et R. Szeliski : High-Quality Video View Interpolation Using a Layered Representation. *In ACM Siggraph*, 2004. 23

Vidéos

Les vidéos suivantes nécessite *Adobe Acrobat Reader 9* et sont également disponibles à l'adresse : http://www.youtube.com/user/petitbenjamin

FIGURE 1 – Film tourné à VRST 2008 durant notre démonstration.

FIGURE 2 – Vidéo présentée à ACM Multimédia 2010 démontrant notre application de téléprésence.

FIGURE 3 – Vidéo présentant un panel de nos applications liées au projet VGate, cette vidéo a été soumise à *Siggraph Emerging Technologies* en 2009 et acceptée pour une démonstration.

FIGURE 4 – Vidéo soumise à CVPR 2011 montrant nos résultats sur plusieurs séquences.